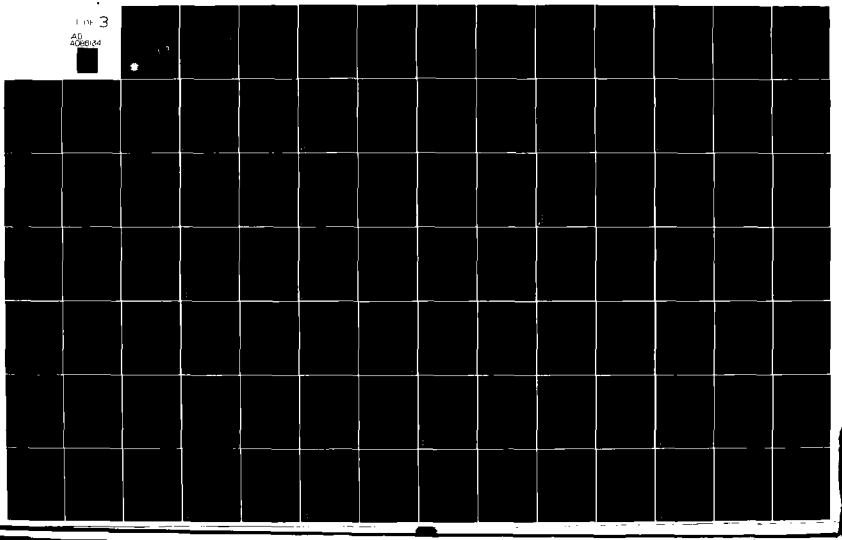
AD-A086 134 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL



LEVEL III

(12)

FINAL REPORT
VOLUME 2

DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 B/S

Department of
ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

(12)

FINAL REPORT
VOLUME 2

DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 B/S

Prepared for

Defense Communications Agency
Defense Communications Engineering Center
1860 Wiehle Avenue
Reston, Virginia 22090

Contract No. DCA 100-79-C-0005

30 April 1980

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 134 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Design and Implementation of a Speech Coding Algorithm at 9600 B/S	5. TYPE OF REPORT & PERIOD COVERED Final Report Nov 1978 - April 1980
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James L. Melsa, et al.	8. CONTRACT OR GRANT NUMBER(s) DCA 100-79-C-0005
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Electrical Engineering University of Notre Dame Notre Dame, IN 46556	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Contract Management Division, Code 260 Washington, D.C. 20305	12. REPORT DATE April 1980
	13. NUMBER OF PAGES 519
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech coding, 9600 bps speech transmission, pitch extraction, adaptive residual coder, waveform reconstruction.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines:

Pitch extraction loop,
Pitch compensating adaptive quantizer,
Sequentially adaptive linear predictor,
Adaptive source coding,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

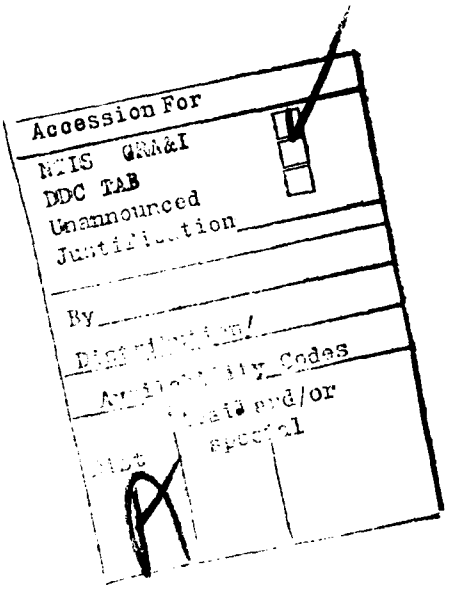
388467

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.



ABSTRACT

This report describes a speech coding algorithm for digital transmission of speech at a rate of 9600 bits per second and the implementation of this algorithm on a speech processing system. The algorithm combines

- Pitch extraction loop
- Pitch compensating adaptive quantizer
- Sequentially adaptive linear predictor
- Adaptive source coding

to generate very high quality speech output. Although each of these elements has been previously applied to speech coding, the combination of all four of these elements has not been studied before. The speech coding algorithm has been implemented on a pair of CSPI MAP 300 Array Processors in real-time in the full-duplex mode.

This report has been bound in two volumes. The first volume contains the narrative description of the algorithm and its development and includes Chapters 1 through 11 and Appendices A through D of the report. The second volume describes the real-time MAP implementation and includes Chapters 12 and Appendices E through G.

PROJECT PERSONNEL

Arvind Arora, research assistant

David L. Cohn, co-principal investigator

James M. Kresse, research assistant

James L. Melsa, principal investigator

Arun K. Pande, research assistant

Maw-lin Yeh, research assistant

FINAL REPORT

DCA CONTRACT 100-79-C-0005

	Page
Abstract	i
Project Personnel	ii
1. Introduction and Outline of Report	1
1.1 Introduction	1
1.2 Summary of Algorithm Requirements	3
1.3 Outline of Report	4
2. Algorithm Description	5
2.1 Introduction	5
2.2 Transmitter Buffer System	11
2.3 Adaptive Low-Pass Filter	16
2.4 Pitch Extraction	18
2.5 Adaptive Residual Coder	20
2.5.1 Adaptive Predictor	20
2.5.2 Adaptive Quantizer	22
2.6 Pitched Repetition	26
2.7 Noiseless Source Coder	28
2.8 Receiver	34
3. Synchronization	38
3.1 Sync. Algorithm Description	40
3.1.1 Synchronization Acquisitions	40
3.1.2 Synchronization Monitor	43
4. Pitch Extraction Studies	46
4.1 Introduction	46
4.2 Pitch Extraction Algorithms	47
4.3 Redundancy Removal	54
4.4 References	61
5. Tree Coding	62
5.1 Introduction	62
5.2 The (M,L) Algorithm	63
5.2.1 Description	63
5.2.2 Results	65
5.3 Adaptive Tree Coding	69
5.3.1 Description	69
5.3.2 Results	71
5.4 Conclusions and Suggestions for Further Research	75
5.5 References	77

6. Backward PARC 78

6.1 Introduction 78
6.2 System Structure 80
6.3 Performance Evaluation and Parametric Studies 85
6.4 Transmission Error Studies 90
6.5 Conclusions 95

7. Tandem Operation 96

7.1 Introduction 96
7.2 PARC in Tandem with CVSD 97
7.3 Effect of Background Noise on PARC Performance 100
7.4 References 103

8. Transmission Errors 104

8.1 Introduction 104
8.2 Simulation of Transmission Error 105
8.3 Minimizing Transmission Error Effects 108
8.4 Conclusion 112
8.5 Reference 113

9. Filtering 114

9.1 Introduction 114
9.2 Evaluation of Pre-emphasis 115
9.3 Filter Selection 118
9.4 Results 121
9.5 Low-Pass Filtering vs. Entropy 123
9.6 Results and Conclusions 125
9.7 References 128
9.A Derivation of Filter 129

10. Buffer Control 136

10.1 Introduction 136
10.2 Overflow control 137
10.3 Underflow control 138
10.4 Special considerations at the receiver 139
10.5 Conclusions and suggestions for further research 140

11. Source and Error Control Coding 141

11.1 Introduction 141
11.2 Source Coding 142
11.2.1 Quantizer levels 142
11.2.2 Side information 144
11.3 Error Control Coding 145
11.4 Conclusion and suggestions for further research 146

A. FORTRAN Simulation of Algorithm 147

B. Segmented SNR Plots 198
B.1 Introduction 198
B.2 Listings 199

C. Plotting Program 242

D. PDP-11 D/A Programs 247

12. MAP Implementation 263

12.1 Introduction 263
12.2 PARC - Transmitter 278
12.2.1 The Pitch Computation Program 281
12.2.2 The Speech Digitization Program 288
12.3 PARC - Receiver 295
12.4 Noiseless Source Coder 302
12.5 Synchronizer, Decoder 311
12.5.1 Synchronization Acquisition Module 313
12.5.2 Decoder Module 315
12.5.3 Initialization Module 318
12.5.4 Decoder Constraints 319
12.6 Program Timing and Speed 320

E. Resampling Program 322
E.1 Operating Details 325

F. CVSD Algorithm 344
F.1 The CVSD System 344
F.2 The Real-Time CVSD System 346
F.2.1 Design Overview 346
F.2.2 The Initialization Step 348
F.2.3 The Processing Step 351
F.3 Conclusion 355
F.4 Reference 355

G. Program Listings for Real Time Implementation of PARC on MAP-300 370

CHAPTER 12
THE REAL-TIME IMPLEMENTATION

12.1 Introduction

In this chapter, the final form of the real-time implementation of the PARC algorithm in a full duplex mode will be presented. The whole implementation will be decomposed into four parts. Each part is described in detail in the following sections. Block diagrams, flow charts and timing diagrams are given to help reader understand the implementation. The program listings are in Appendix G.

Before presenting the implementation, the system components used will be described. A MAP-300 (Macro Arithmetic Processor) produced by Computer Signal Processing, Inc. is used to implement the speech coding algorithm (see Table 12.1). A host computer and MAP-300 system block diagram is shown in Fig. 12.1. The MAP-300 consists of six programmable processing elements as well as memory and peripherals:

A. A Central Processing Unit (CSPU) functions as the executive controller by interpreting commands from the host computer, setting up and scheduling the other processors, and performing arithmetic and logic operations.

B. An Arithmetic Processor (AP) consists of two subprocessors. An Arithmetic Processing Unit (APU) carries out the floating point arithmetic calculations. An Arithmetic Processing Scroll (APS) is a data addressing device for the APU.

C. An Input/Output Scroll (IOS-2) transfers bit streams into or out of a modem.

D. An Analog Data Acquisition Module (ADAM) samples and quantizes input analog signals into digital signals and stores them into main memory.

Table 12.1
MAP-300 Speech Processing System

Item No.	Qty.	Model Number	Description
1	1	1030	MAP-300 Processor
2	1	2030	8K x 32 MOS Memory
3	1	2050	16K x 32 MOS Memory
4	1	2203	8K x 32 MOS Memory
5	1	2120	2K x 32 Bipolar Memory
6	1	3100	PDP-11 Interface
7	1	4020	Model 2SM/ I/O Scroll
8	2	4040	Bus Switch
9	1	5120	Analog Data Acquisition Module
10	1	5130	Analog Output Module
11	1	6100	Expansion Chassis
12	1	6200	Auxiliary Power Supply
13	1	03-1360585	GTE Speech Processing Interface

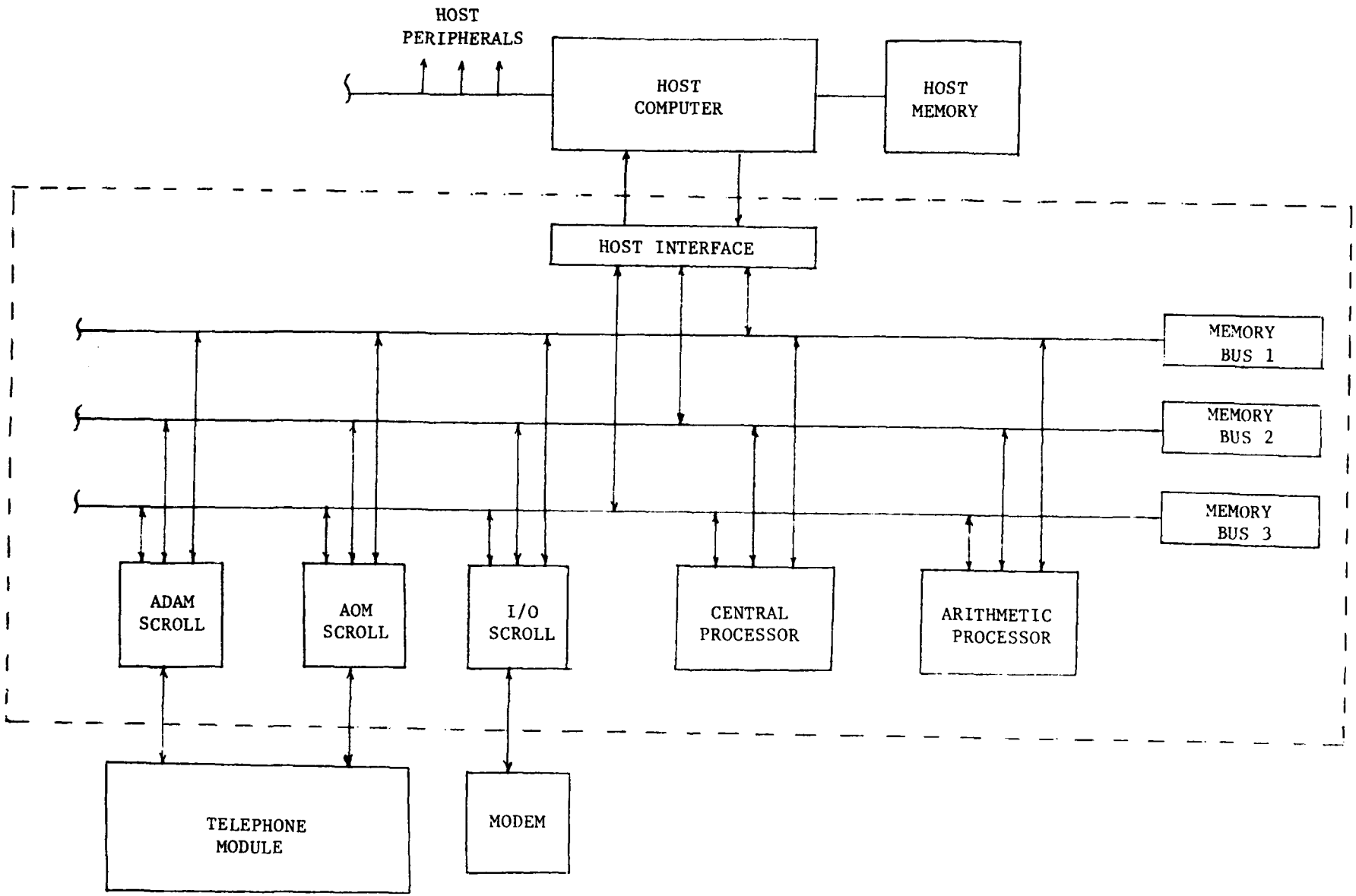


Fig. 12.1 MAP-300 System Block Diagram

E. An Analog Output Module (AOM) converts input digital signals into analog signals and outputs them to a telephone module.

F. A Host Interface Scroll (HIS) acts as a communication and data transfer medium between the host computer and the MAP-300 system.

This array processor system provides two classes of software. The first class is the software designed specifically for array processing. It is an executive-driver-subroutine package resident in the MAP-300 and in the host computer which permits direct calling of routines to perform arithmetic operation, to manage MAP memory, to define even higher level routines. A MAP user can execute array processing operations with simple Fortran calls in the host computer. The sophisticated programmer even can implement his own processing algorithms in MAP assembly languages. To implement the PARC digitization system, some new array and non-array functions, which are described later, are employed. The second class consists of utility programs including a cross-assembler and a cross-simulator. These may be used to add new routines to the array processing library.

The array processor can be used as a peripheral to a host computer, using host peripherals for data storage and interface for data and commands transfer between them. However, after initialization, it can stand alone by using proper control functions in the host.

The underlying design principles of the real-time implementation are as follows: First, in order to study system performance, the flexibility of changing system parameters such as speech algorithm parameters, buffer sizes and buffer starting locations must be provided. Second, in order to be in the real-time mode, the arithmetic execution time and the overhead time have to be reduced. The arithmetic execution time can be

decreased by efficiently utilizing the arithmetic processors. The overead time can be decreased by utilizing all processors as asynchronously and as simultaneously as possible. However, the necessary synchronization between processors has to be carefully considered.

After several months of study, the final real-time PARC communication system consists of the following programs:

1. Host programs: Fortran PARC main program
   Fortran PARC host support program
2. AP programs: PARC-transmitter pitch computation program
   PARC-transmitter speech digitization program
   PARC-receiver
3. CSPU programs: Encoder program
   Decoder program
   Synchronization program
   Transmitter buffer pointers update program
   Receiver buffer pointers update program
4. IOS programs: ADAM program
   ADM program
   IOS-2 program

The main function of the host Fortran programs is to initialize the MAP-300 system with the real-time PARC algorithm, to set up the proper command sequences and to initiate them. After that, the MAP-300 system can execute the PARC without any further commands from the host computer. The two PARC-transmitter programs and the PARC-receiver program require sophisticated arithmetic operations and will be executed in the AP. Those programs requiring logic and bit operations include encoder, decoder, synchronizer and address pointers update programs. These will

be executed in the CSPU. A/D converting, D/A converting and bit-transfer will be executed in the ADAM, the AOM and the IOS-2 respectively. These five processors operate in parallel. The host Fortran modules will be described below; they present the whole picture of the PARC. The rest of the programs will be presented in the following sections.

The flow chart of the real-time PARC main program is shown in the Fig. 12.2. The module can be divided into two steps: An initialization step and a processing step. In each step, each processor has its own role.

1. The initialization step:

In this step, the host computer will initialize the whole algorithm by reading in system parameters, configuring logical buffers, initializing logical buffers, loading ADAM and AOM, and creating appropriate command sequences called function lists. The logical buffers and the function lists, which are fairly complicated, will be described later. In this step, only three processors in the MAP-300 system have been used as follows:

A. The CSPU acts as the resource controller. Responding to a host command, it loads an A/D program to the ADAM, loads a D/A program to the AOM, configures the logical buffers, and loads appropriate arithmetic functions into the AP.

B. The AP executes these arithmetic functions which will reset the logical buffers.

C. The HIS acts as an interface between the host computer and the MAP-300.

The Fortran host support program, which is the host support module for those new array and non-array functions in the real-time PARC

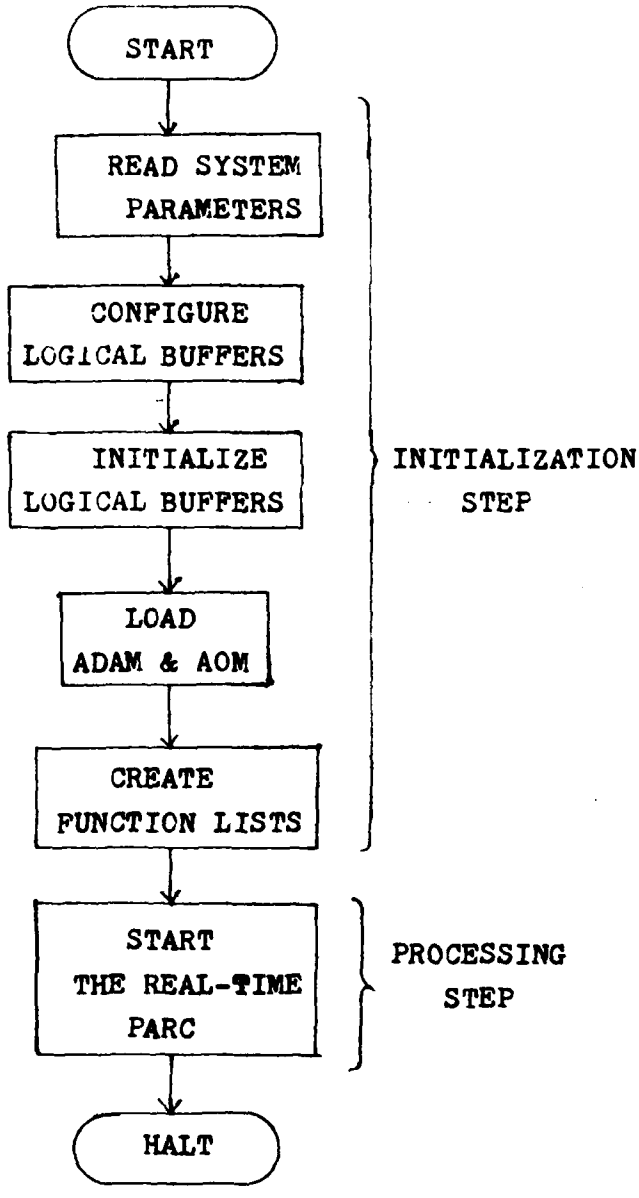


Fig. 12.2 **The Flow Chart of the Main Program**

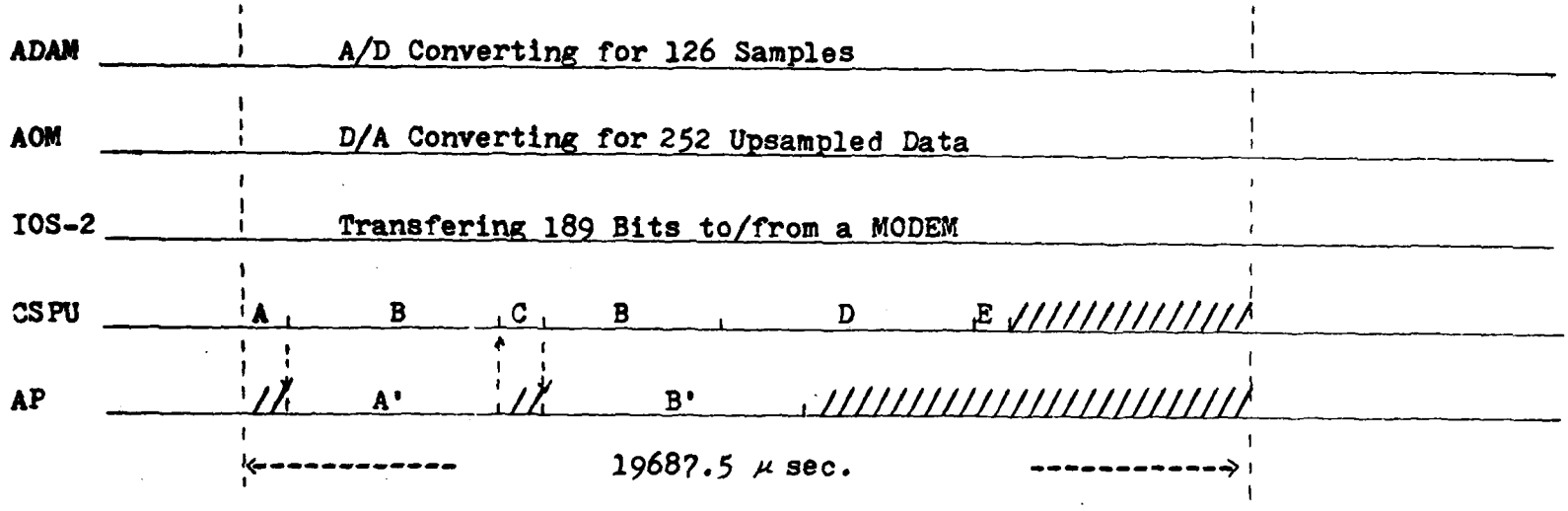
algorithm, interfaces the main program to the host/MAP driver module. The main functions of this support program are as follows:

1) Check each argument for the proper range and return error code.
2) Pack arguments for the MAP in a Function Control Block (FCB)
3) Determine any host absolute addresses that must be evaluated while constructing the Data Control Block (DCB).
4) Pass DCB to the host/MAP driver.

2. The processing step:

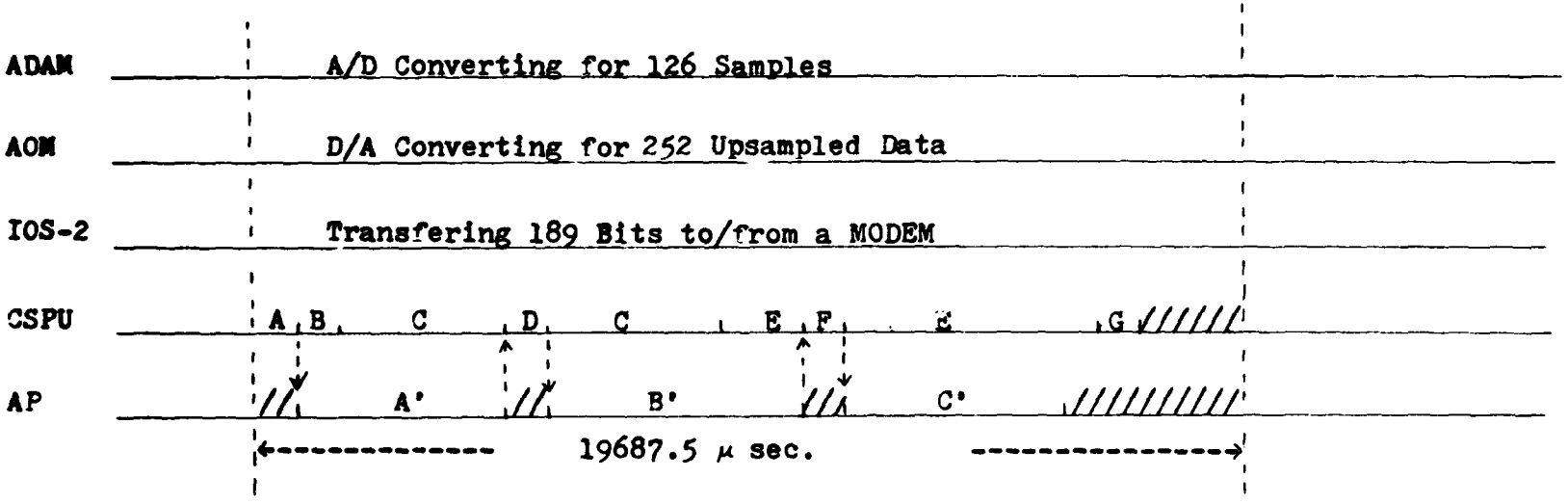
After initialization, the MAP-300 can execute the real time PARC algorithm without any further commands from the host computer. Because of the fixed delay for the whole system (refer to Section 2.2), a fixed time slot has been chosen as a main parameter to synchronize all the processors in the MAP-300 system as shown in Fig. 12.3 and Fig. 12.4. In each time slot, a particular function list, which is described below, will be executed. This fixed time slot is set to be 19687.5 microseconds which is precisely the period for ADAM to process 126 samples, the period for AOM to process 252 upsampled data, and the period for IOS-2 to process 189 bits.

The function lists, whose relationships are shown in Fig. 12.5, are eight sets of command sequences. When the real-time PARC is initiated, the first function list, which starts the ADAM, the AOM and the IOS-2, is executed. Afterward, the main control function list (the second function list) is executed by the CSPU. This function list assures the synchronization of the receiver. The execution of the list is as follows:



A: Load the APU and the APS with the Pitch Computation Program
B: Encoder
C: Load the APU and the APS with the Speech Digitization Program
D: The Channel Synchronization Program
E: Update the PARC-Transmitter Address Pointers
A': The Pitch Computation Program
B': The Speech Digitization Program

Fig. 12.3 The Timing Diagram for the Synchronization Operation



A: Load the APU and the APS with the Pitch Computation Program
B: Update the PARC-Receiver Address Pointers
C: Encoder
D: Load the APU and the APS with the Speech Digitization Program
E: Decoder
F: Load the APU and the APS with the PARC-Receiver Program
G: Update the PARC-transmitter Address Pointers
A': The Pitch Computation Program
B': The Speech Digitization Program
C': The PARC-Receiver Program

Fig. 12.4 The Timing Diagram for the Normal PARC Operation

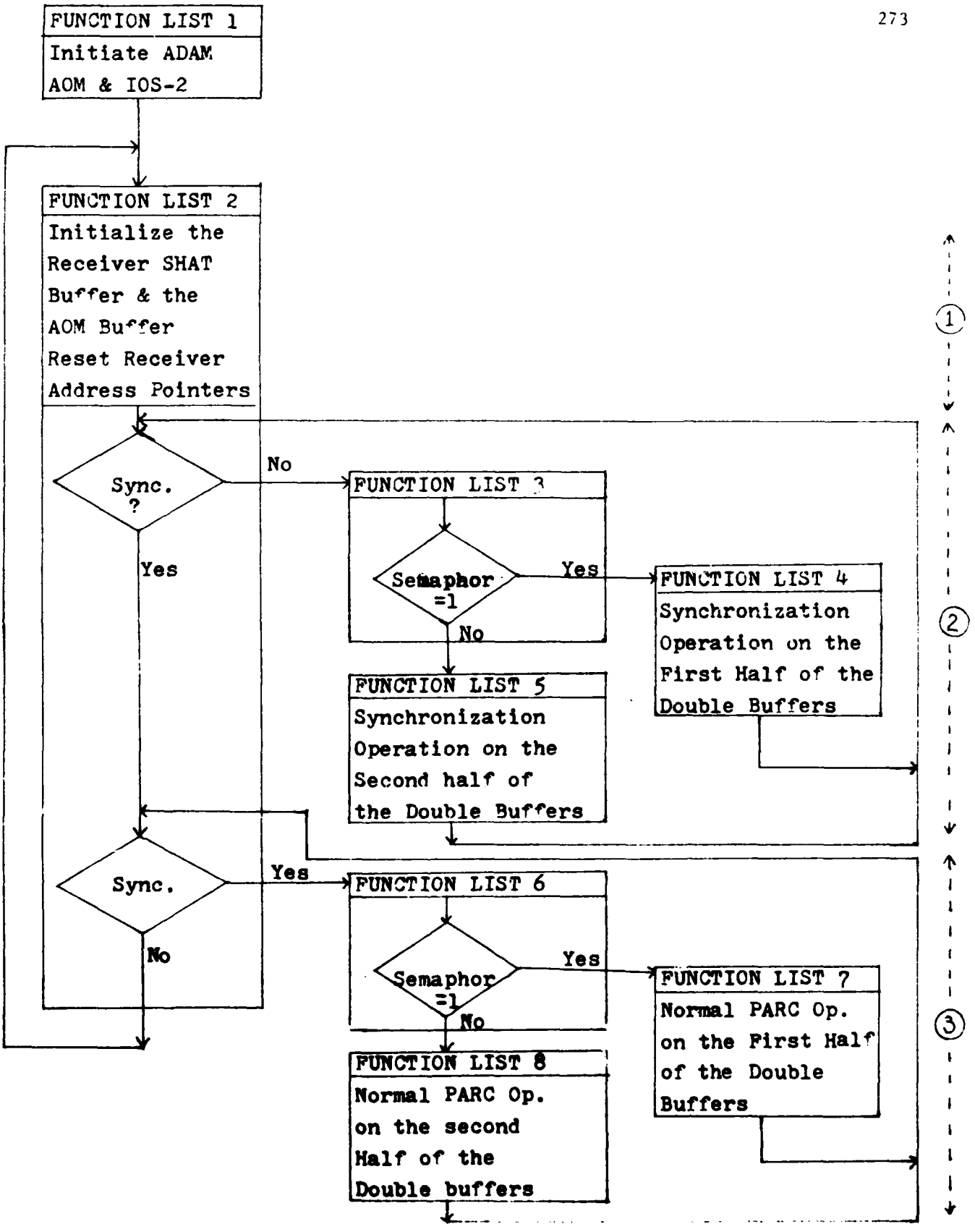


Fig. 12.5 The Flow Chart of Function Lists

1. It initializes the receiver SHAT buffer and the AOM buffer. Also, it resets the receiver address pointers.
2. If the channel synchronization does not exist, it keeps executing the synchronization operation until a channel synchronization is set. The synchronization operation consists of the PARC-transmitter, the encoder and the synchronizer.
3. If the channel synchronization exists, it keeps executing the normal PARC operation until a channel synchronization is lost. The normal PARC operation consists of the PARC-transmitter, the encoder, the decoder and the PARC-receiver.
4. Go to 1.

Because of parallel data processing which speeds up the whole system (refer to the Section 2.2), the double buffering technique is employed. Therefore, the synchronization operation and the normal PARC operation each consists of three function lists: A control function list, a function list which operates on the first half of the double buffers and a function list which operates on the second half of the double buffers. The control function list checks a semaphore which resides in the MAP memory and then executes a function list which will operate on the proper buffers. The reason for this is to have a continuity of data flow for the PARC-transmitter. For instance, suppose the receiver loses the channel synchronization after the normal PARC operation on the first half of the double buffers. Then the main control function list executes the synchronization operation. The control function of the synchronization operation checks the semaphore and then executes the synchronization operation on the second half of the double buffers. So, the transmitter

gets a continuous data flow. The main control function list will keep executing until the MAP-300 system is terminated.

In the synchronization operation, which consists of the function lists 3, 4 and 5 (shown in Fig. 12.3), the role of each processor in a time slot is described as follows:

1) The ADAM samples and quantizes the input analog signal from the telephone module and stores these samples into MAP main memory. In a time slot, it produces 126 quantized samples.
2) The AOM acts as a D/A converter and outputs 252 silence samples since no information flows into the receiver.
3) The IOS-2 acts as an input/output interface between the MAP-300 and the channel modem. In a time slot, it transfers 189 bits out of the encoder bit buffer and reads 189 bits into the decoder bit buffer.
4) The CSPU functions as follows in sequence:
   A: It loads the AP with the pitch computation program.
   B: It executes the encoder. However, upon request from an APU interrupt, it suspends the current operation and
   C: Loads the AP with the speech digitization program. It restarts the encoder after the loading.
   D. It executes the channel synchronization program.
   E. It updates the PARC-transmitter address pointers.
5) The AP executes the arithmetic part of the PARC-transmitter. After finishing each program, it will interrupt the CSPU to indicate that it is free for the next operation, it any. In this operation, the AP has two jobs:

A: The pitch computation program.

B: The speech digitization program.

These five processors function in parallel. The ADAM, AOM and IOS-2 operate continuously without any interrupt. However, the CSPU and the AP have to finish their own job inside the time slot in order to be in real-time.

The normal PARC operation (shown in Fig. 12.4) which consists of the function lists 6, 7 and 8, has the same ADAM and IOS-2 operation as the synchronization operation. However, the CSPU, the AOM and the AP have different operations as follows:

1) The ADAM samples and quantizes the input analog signal from telephone module and stores these samples into MAP main memory. In a time slot, it produces 126 quantized samples.
2) The AOM acts as a D/A converter and outputs 252 unsampled reconstructed speech samples to the telephone module.
3) The IOS-2 acts as an input/output interface between the MAP-300 and the channel modem. In a time slot, it transfers 189 bits out of the encoder bit buffer and reads 189 bits into the decoder bit buffer.
4) The CSPU functions as follows in sequence:
   A: It loads the AP with the pitch computation program.
   B: It updates the receiver address pointers.
   C: It executes the encoder. However, upon request from an APU interrupt, it suspends the current operation and
   D: Loads the AP with the speech digitization program. It restarts the encoder after the loading.

E. It executes the decoder. Upon request from an APU interrupt, it suspends the current operation and

F. Loads the AP with the PARC-receiver program. It restarts the decoder after the loading.

G. It updates the transmitter address pointers,

5) The AP executes the PARC-transmitter and the PARC-receiver. After finishing each program, it will interrupt the CSPU to indicate that it is free for the next processing, if any. In this operation, the AP has three jobs:

A: The pitch computation program.

B: The speech digitization program.

C: The PARC-receiver program.

In this operation, the timing control is very important which will be discussed in the next section.

The subsequent sections will describe each of the subsystems in the real-time PARC system. The next section explains the design of the PARC-transmitter. The complicated buffer system, buffer control and the speech digitization will also be described. The corresponding noiseless source encoder will be explained in Section 12.4.

Section 12.3 describes the design of the PARC-receiver. Although some parts of the system are the same as in the transmitter, the different design of the buffering and upsampling will be depicted. The corresponding noiseless source decoder will be explained in Section 3.5.

12.2 The PARC-Transmitter

In this section, the main line of the real-time PARC-Transmitter, which resides in the AP, will be presented. The corresponding encoder will be discussed in Section 12.4. The block diagram of the PARC algorithm is shown in Fig. 2.2 and discussed in Chapter 2.

Because of the limit on the size of the AP memory, the whole PARC-Transmitter algorithm cannot be implemented as a single AP program. So, the PARC-Transmitter has to be cut into two subprograms, namely, the pitch computation program and the speech digitization program, as shown in Fig. 12.6. Those portions of the algorithm which operate on blocks of speech samples will be put in the pitch computation program. Those portions include: input gain factor calculation, system noise reducer, adaptive filter and pitch extractor. These elements of the algorithm which process speech sample-by-sample will be combined into the speech digitization program. These elements are the adaptive quantizer, the inverse quantizer, the adaptive predictor and the pitch extraction.

Before any further description of the design of this PARC-Transmitter, the buffer system has to be depicted. There are seven buffers employed by the PARC-Transmitter as shown in Table 12.2.

Among them, ADAM, LEVEL and BIT buffers are double buffers which allow one processor to fill one half of the buffer while another processor processes on the other half. The SAMPLE, VHAT and SHAT buffers are circular buffers which allow continuous access. However, the additional address pointers, which indicate the starting locations of those circular buffers at the beginning of a time slot, have to be considered. The PARAMETER buffer is a single buffer which stores of system parameters such as predictor coefficients, quantizer output

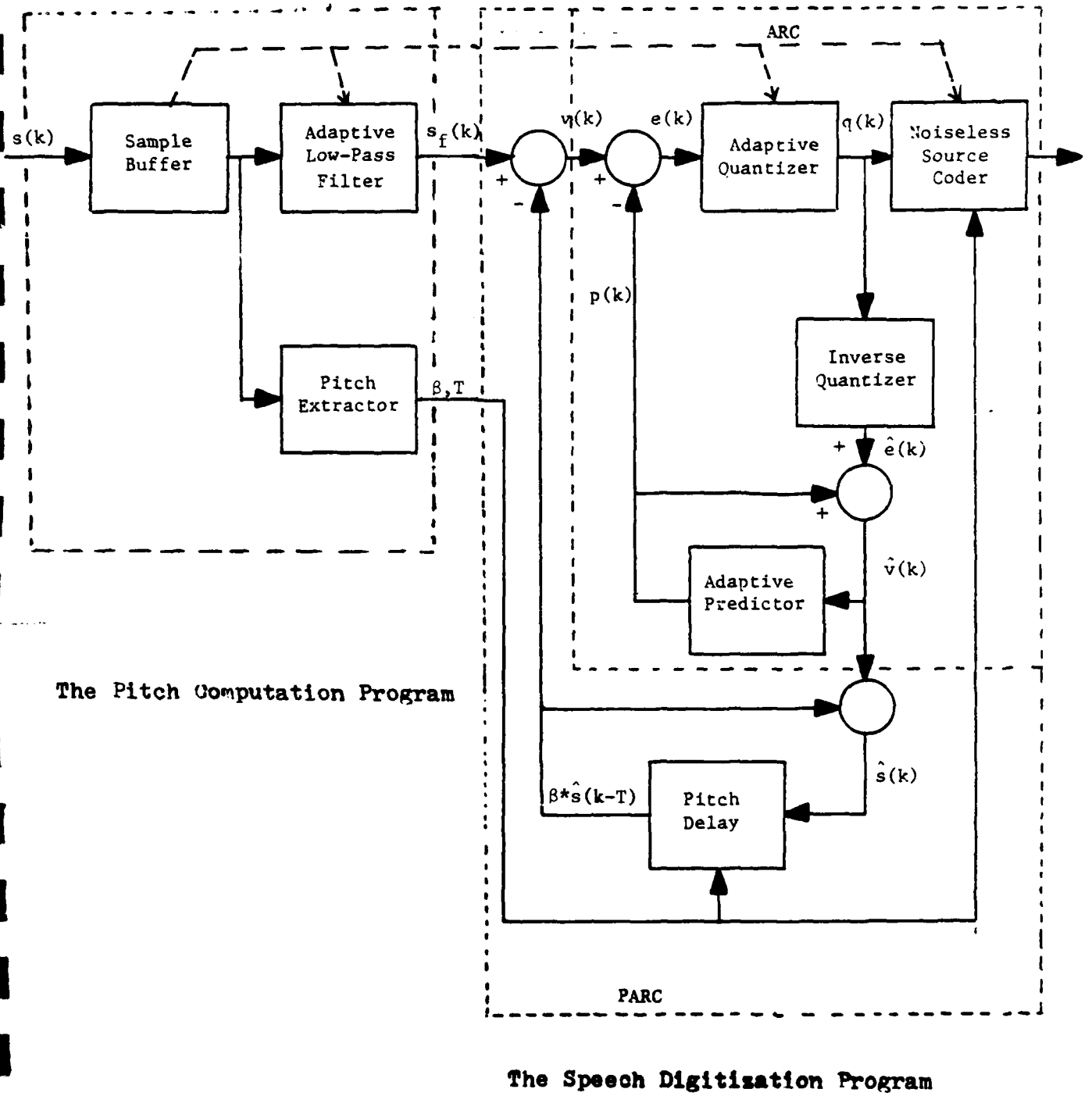


Fig. 12.6 THE BLOCK DIAGRAM OF THE PARC

Table 12.2 Buffer Configuration in the PARC-Transmitter

Name	Bus #	Starting Location	Ending Location	Buffer Size	Buffer Type
ADAT	3	3072 3198	3197 3323	126 126	Double Buffers, Short Floating Point
SAMPLE	3	0	1023	1024	Circular Buffer, Short Floating Point
VHAT	2	0	1023	1024	Circular Buffer, Short Floating Point
SHAT	3	1024	2047	1024	Circular Buffer, Short Floating Point
PARAMETER	2	3072	3331	20	Single Buffer, Long Floating Point
LEVEL	1	25000 27000	25599 27599	600 600	Double Buffer, Long Fixed Point
BIT	1	25700 27700	25899 27899	200 200	Double Buffers, Long Fixed Point

scaling factors, quantizer expansion factors, and the number of bits associated to each quantizer level.

The PARC-Transmitter employs four processors: The ADAM, the AP, the CSPU and the IOS-2. In order to achieve parallel processing, the double buffering technique is used. The function of these buffers associated to the processors at a time slot is shown in Fig. 12.7. At a time slot, the following processes are executed in parallel:

1. While the ADAM is filling a half of the ADAM buffer, the AP is accessing the other half.
2. While the AP is filling a half of LEVEL buffer, the CSPU is accessing the other half.
3. While the CSPU is filling a half of the BIT buffer, the IOS-2 is accessing the other half.

In the next two subsections, the pitch computation program and the speech digitization program will be described. The complicated buffer controller will also be presented.

12.2.1 The Pitch Computation Program

The pitch computation program contains the gain controller, a noise reducer, a buffer controller, the adaptive filter and the pitch extractor. The APU flow chart and the corresponding APS flow chart, including the controlling flows are depicted in Figs. 12.8 & 12.9. The detail functions are described as follows and the parallel processing is shown:

1. In response to the function list, the CSPU loads the APU and the APS modules of the pitch computation program into the APU and the APS respectively.
2. The CSPU then sets flag RI which will initiate the APS module.

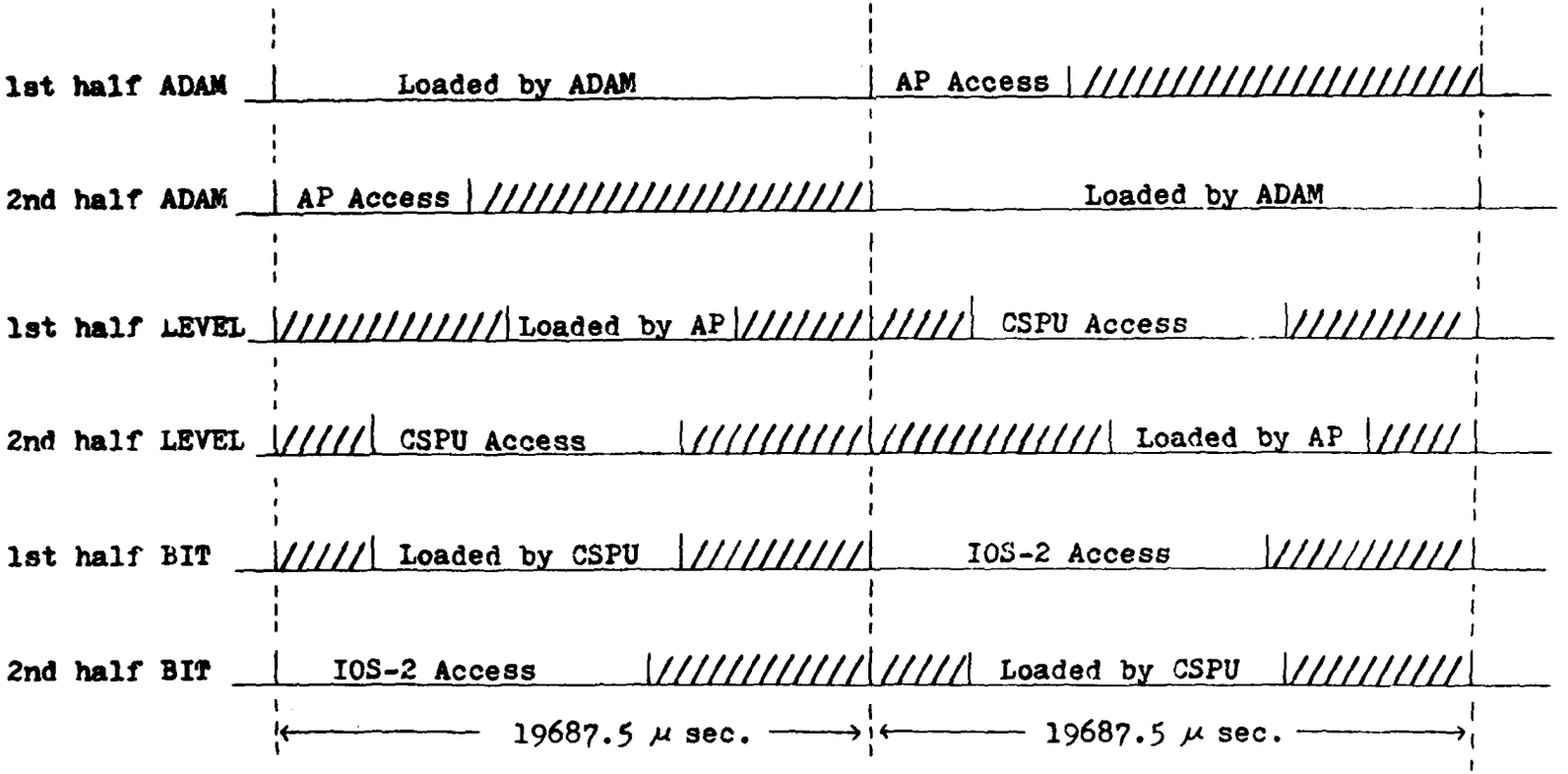


Fig. 12.7 The Double Buffering in the PARC-transmitter

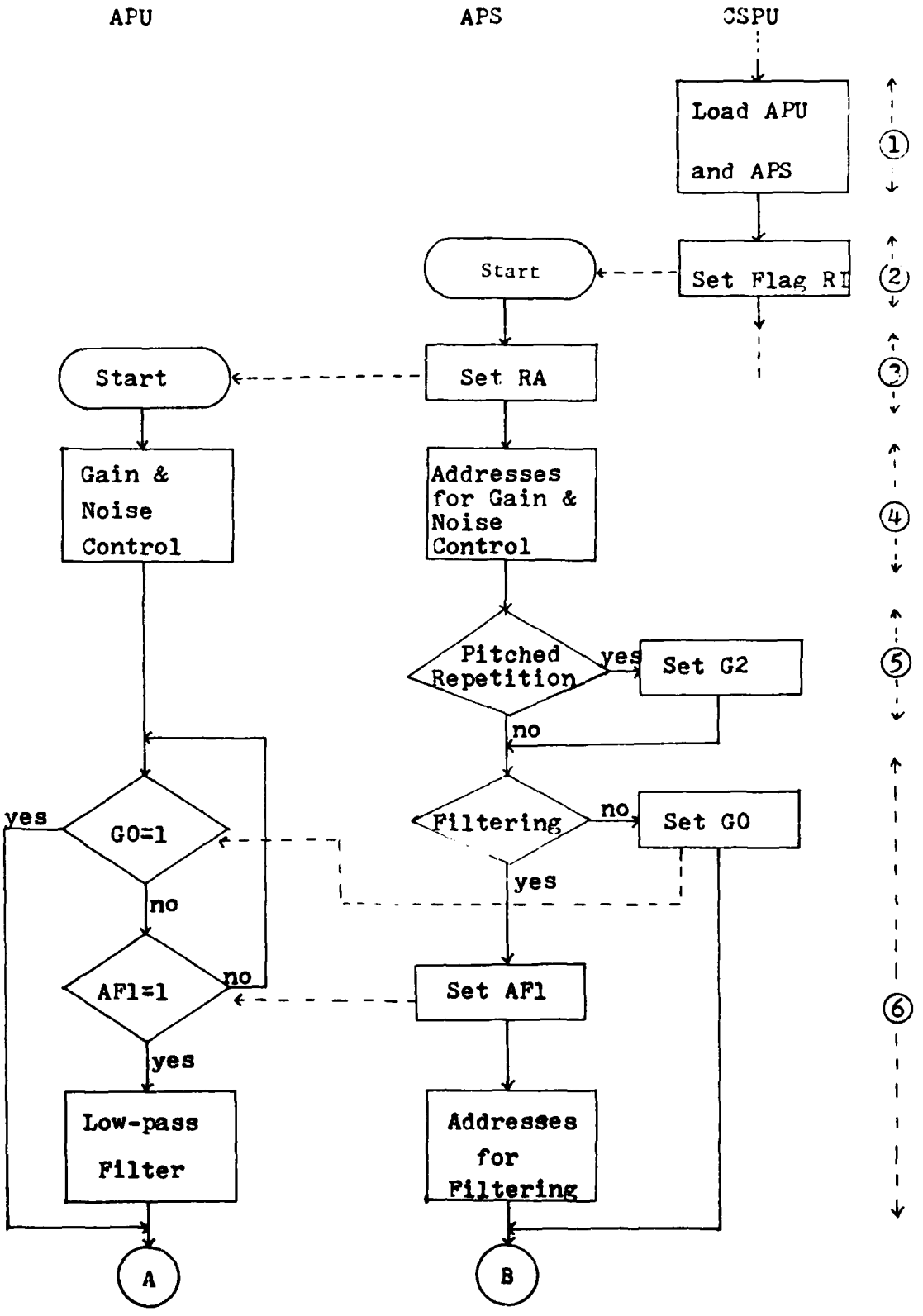


Fig. 12.8a The Flow Chart of the Pitch Computation Program

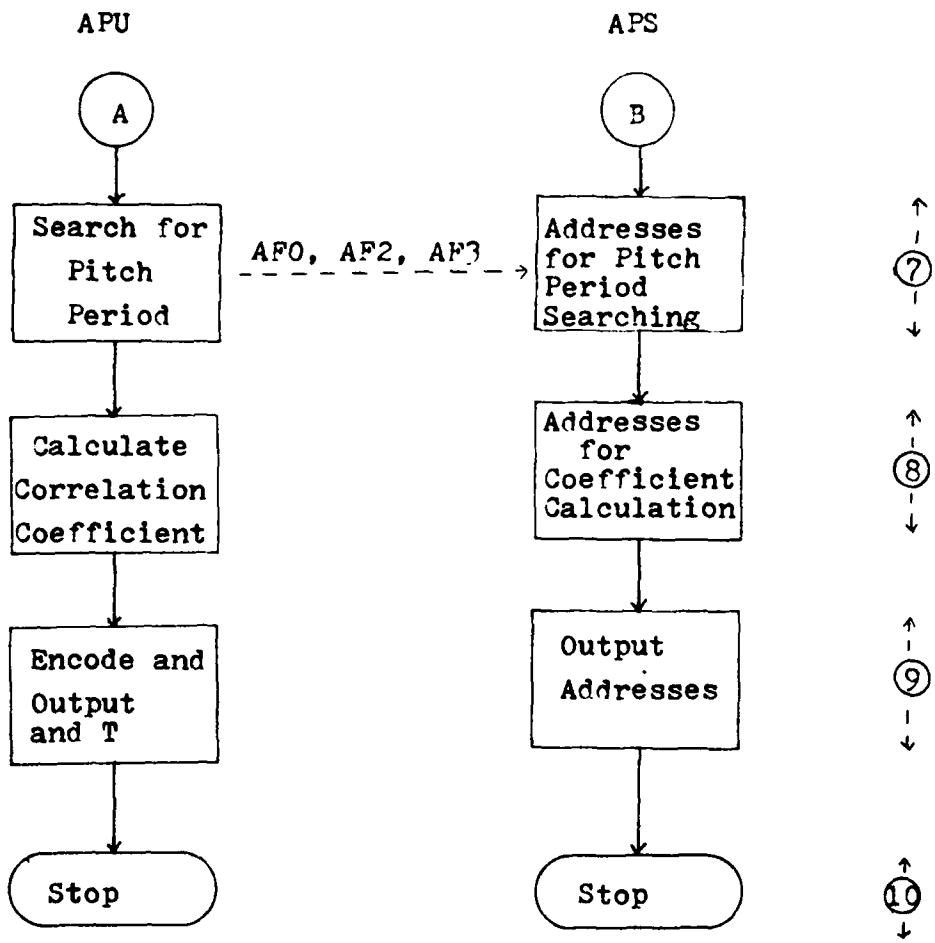


Fig. 12.8b **The Flow Chart of the Pitch Computation Program**

3. The APS module sets flag RA which will start the APU module.
4. The APS produces the addresses of the noise reducing parameter, the gain factor, 126 input data addresses from the ADAM buffer and 126 output data addresses to the SAMPLE buffer.

   The APU reads 126 input samples whose addresses are given by the APS, reduces their noise, multiplies by the gain factor and outputs to the SAMPLE buffer. The input samples from the ADAM are 11-bit precision plus a sign bit in a range of [2047/2048,-2048/2048]. After being read by the AP, these are converted into 32-bit floating point numbers.
5. The APS checks the fullness of the SAMPLE buffer, sets the flag G2 if it is over the pitch repetition threshold.
6. The APS again checks the fullness of the SAMPLE buffer. If it is under the filtering threshold, the APS sets the flag GO and jumps to Step 7. If it is over the threshold, the APS sets the flag AF1 and produces addresses for the filtering. The APU waits for the signals from the APS. If the flag GO is set, it goes to Step 7. Otherwise, it executes the filtering process which is described in Section 2.3.

7. The APS produces addresses for the data used to compute the pitch period. The APU computes the pitch period using the AMDF technique (refer to Section 2.4). In order to speed up the whole real-time system, all processors have to be utilized as asynchronously and as simultaneously as possible., i.e., the minimum use of the communication flags between the APU and the APS. However, because of the limit on the number of the registers in the APS, three flags are used in this portion to search for the pitch period as shown in Fig. 3.9. In this block, the APU determines how many iterations are left to be executed. The APS has to wait for the information, the flags AF0 and AF1, at the end of each iteration.
8. The APS produces addresses for the data used to calculate the pitch correlation coefficient. If the flag G2, which indicates the pitched repetition, is set, it produces addresses starting a pitched repetition block behind. The APU calculates the pitch correlation coefficient.
9. The APS produces output addresses. The APU encodes the pitch period and the pitch correlation coefficient and outputs them to the MAP memory.
10. The AP interrupts the CSPU to indicate that it is finished.

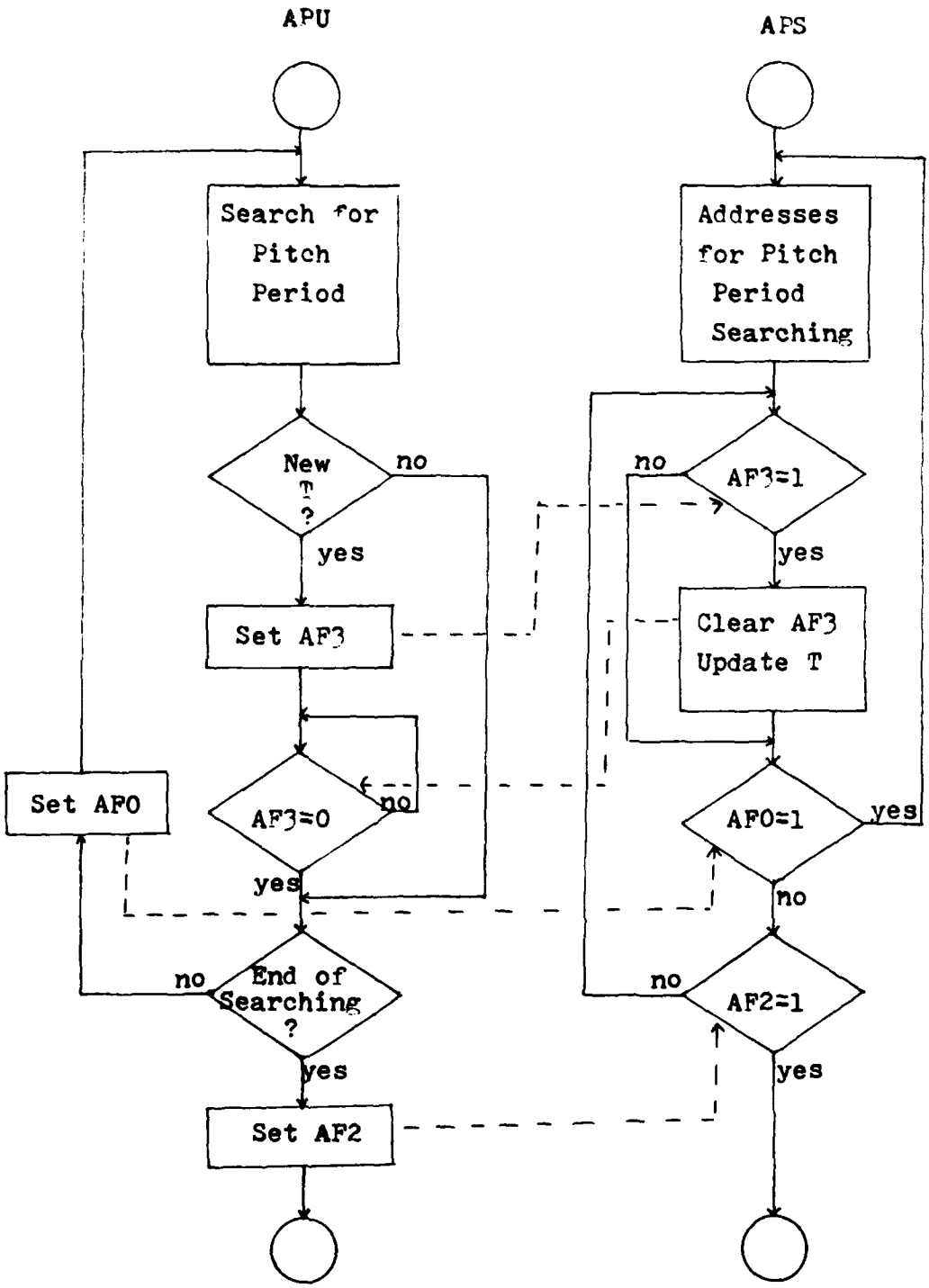


Fig. 12.9 The Flow Chart of the Pitch Period Searching Loop

12.2.2 The Speech Digitization Program

The speech digitization program contains the pitched repetition, the adaptive predictor, a maximum number of processed samples controller, a reciprocal function, the pitch extraction, the adaptive quantizer, the inverse adaptive quantizer, the bit buffer controller, the underflow controller, and a parameter update. The APU and the APS flow chart including the control flows are depicted in the Fig. 12.10 because the number of samples which are processed by the transmitter varies in each time slot, the flags AFO, AF3 and G1 are used to communicate between the APU and the APS. The flag APO, which is controlled by the APU, is used to indicate the beginning of the digitization loop. The flag G1, which is controlled by the APU and the APS, is used to indicate an underflow of the sample biffer or reaching the maximum number of samples permitted in the time slot. The flag AF3, which is controlled by the APU, is used to indicate that at least 157 information bits have been generated.

The detail functions are as follows:

1. In response to the function list, the CSPU loads the APU and the APS modules of the speech digitization program into the APU and the APS respectively.
2. The CSPU then sets the flag RI which will initiate the APS module.
3. The APS module sets the flag RA which will start the APU module.
4. The APS produces the addresses of the system counters. The APU reads in the system counters. The system counters are the BIT counter, SAMPLE counter and the LEVEL counter. Because the fixed time slot is used as the main parameter for the

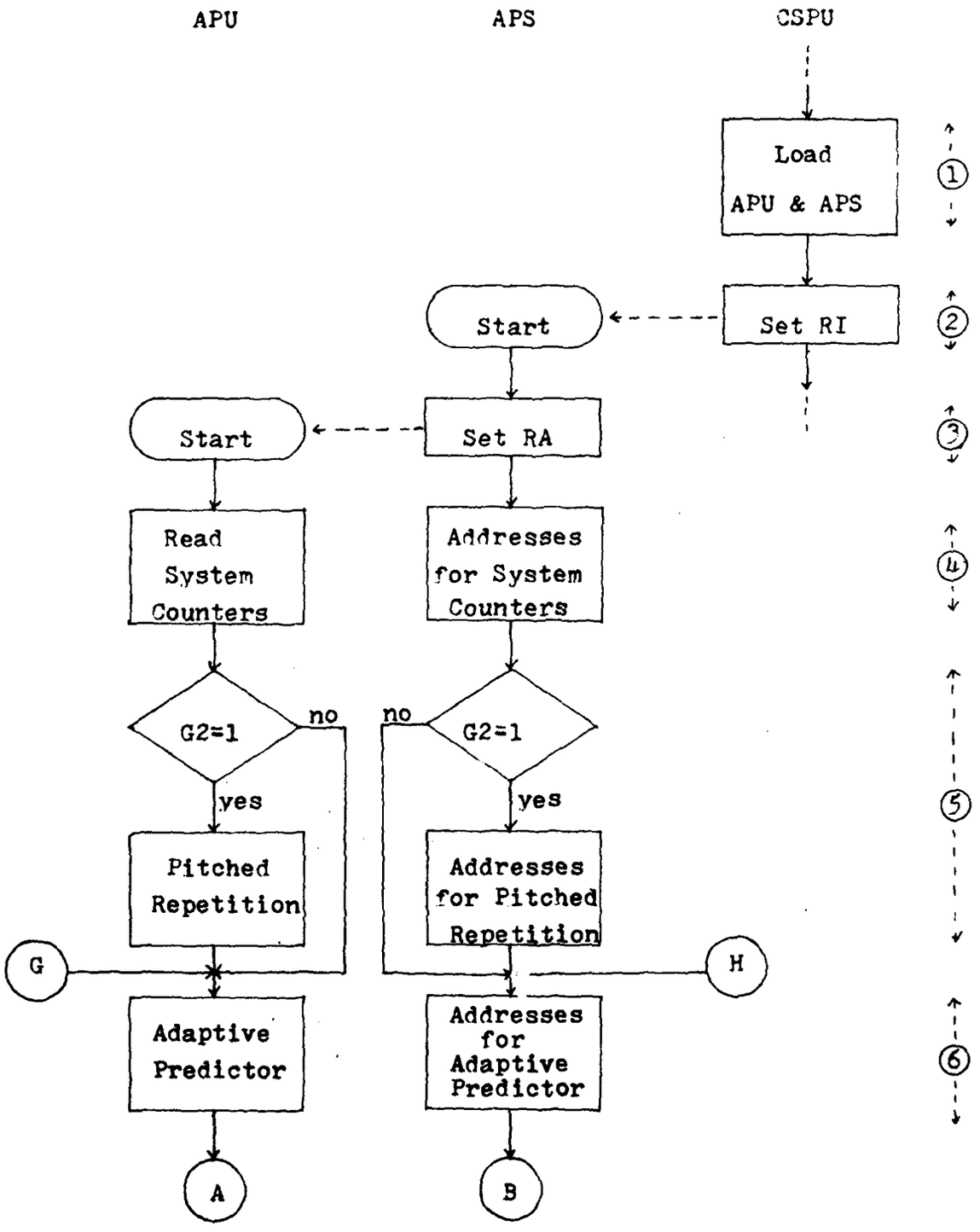


Fig. 12.10a **The Flow Chart of the Speech Digitization Program**

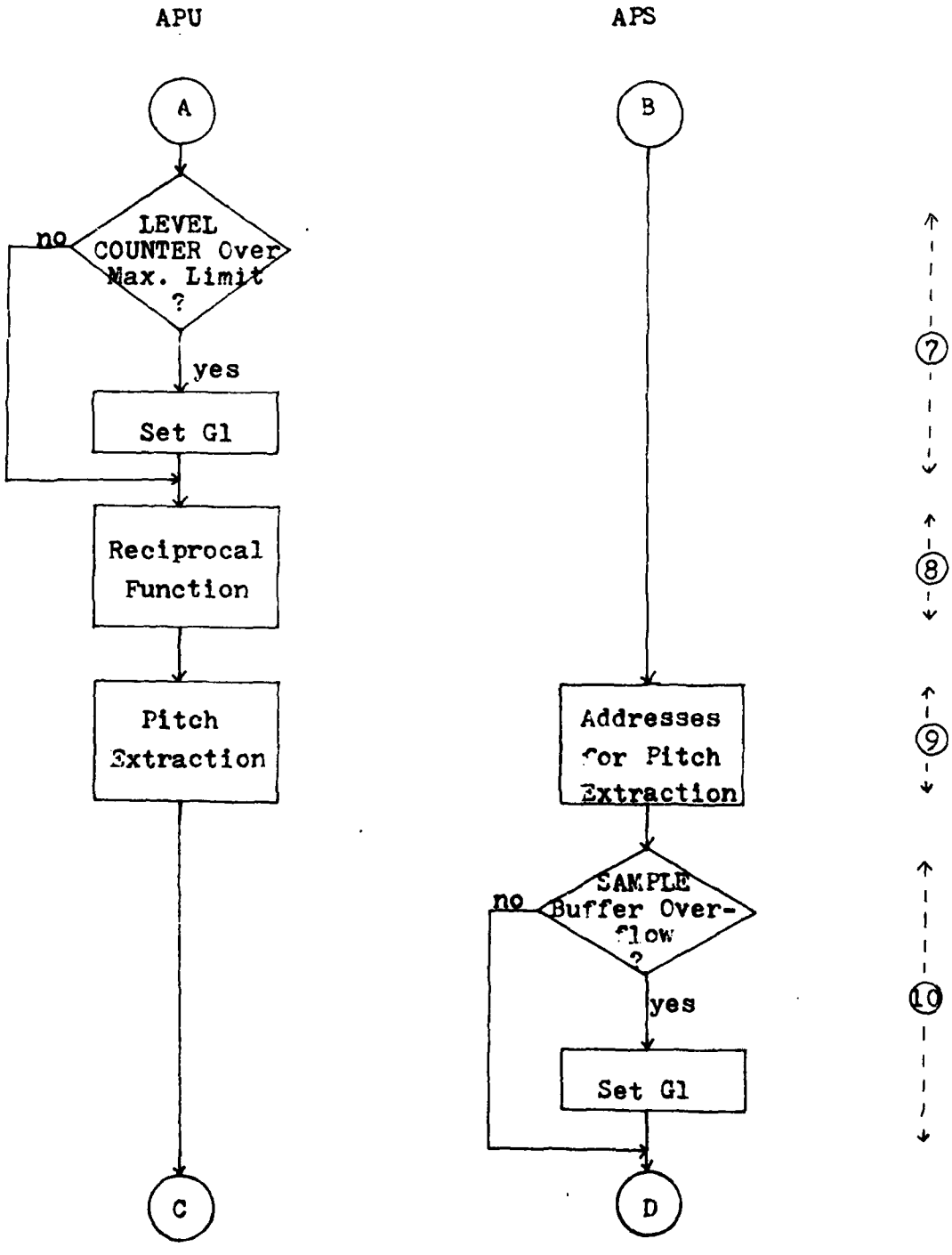


Fig. 12.10b The Flow Chart of the Speech Digitization Program

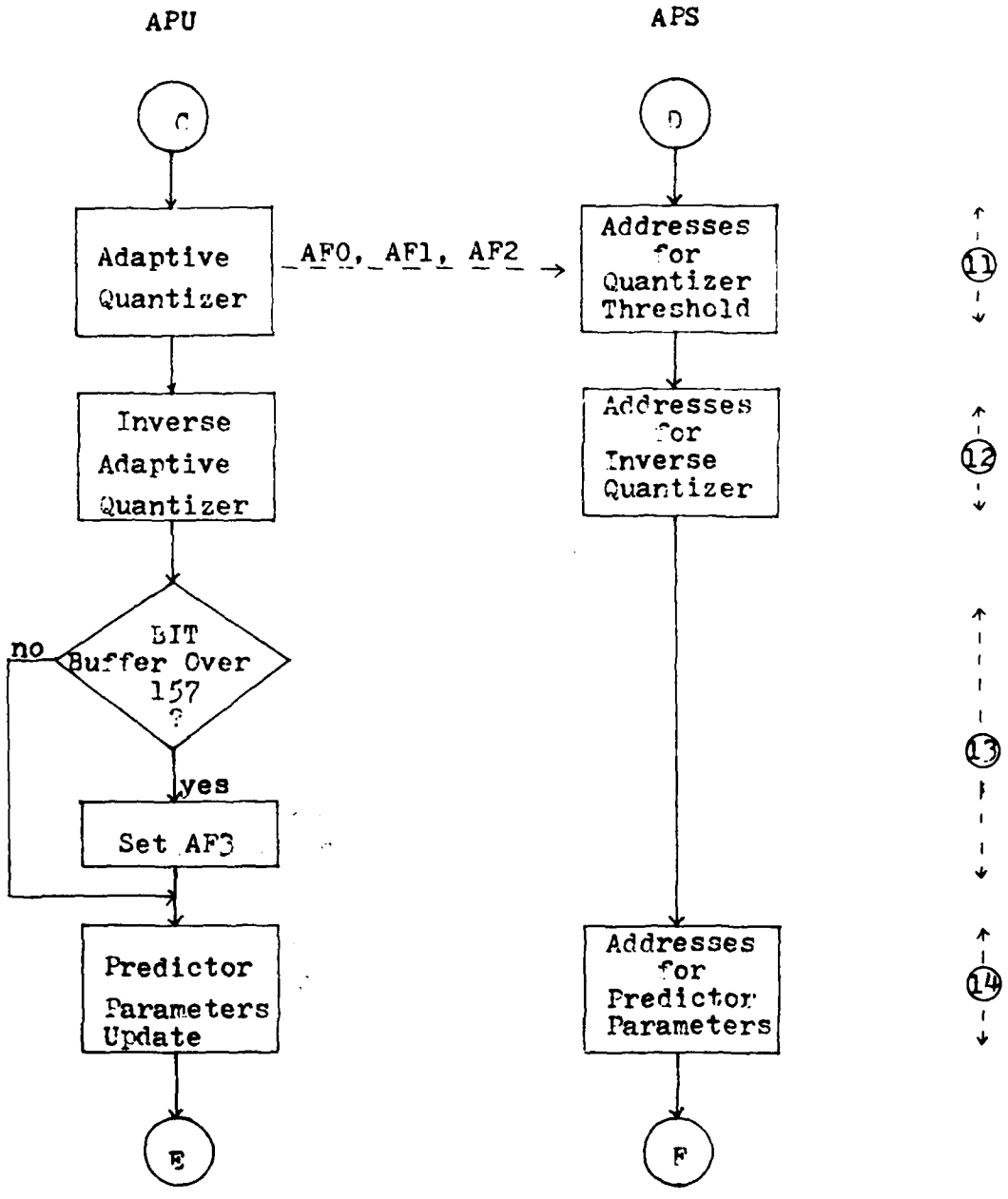


Fig. 12.10c The Flow Chart of the Speech Digitization Program

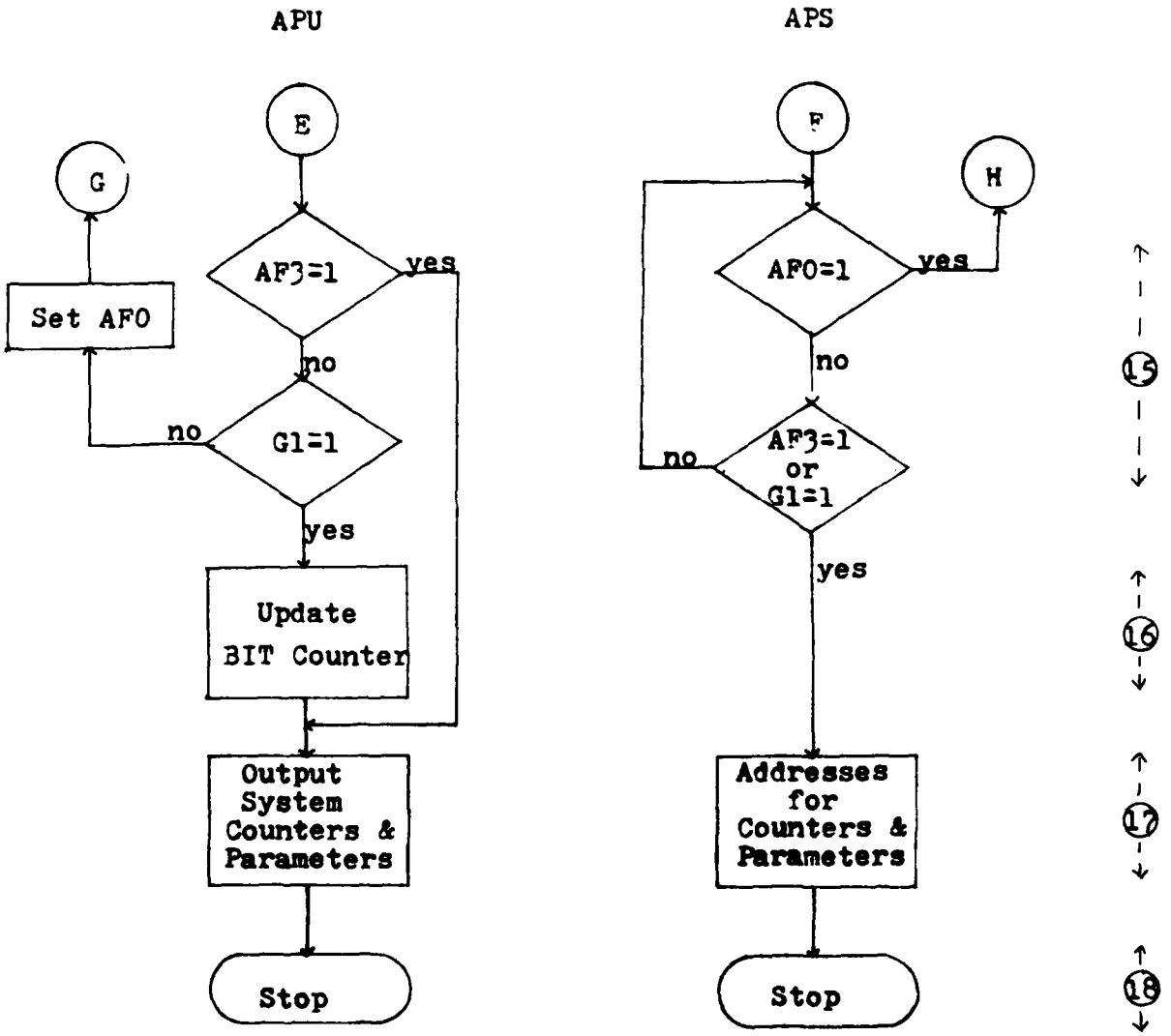


Fig. 12.10d The Flow Chart of the Speech Digitization Program

real-time system, the IOS-2 transfers 189 bits in a time slot. However, as described in Chapter 2, only 157 bits are available to encode the quantizer levels. the BIT counter is used to control the PARC-Transmitter so as to generate just more than 157 information bits. The SAMPLE counter is used to count the total number of samples processed by the PARC-Transmitter during a time slot. The address pointer update program will use this information to update the address pointers in the transmitter circular buffers. The LEVEL counter is used to limit the maximum number of samples which can be processed in a time slot. This assures real-time operation.

5. The APU and the APS check the flag G2 which is controlled by the pitch computation program to indicate the condition of the pitched repetition. If flag G2 is clear, there is no pitched repetition and the APU and the APS go to Step 6. Otherwise, the APS produces the addresses of SHAT buffer for the pitched repetition. The APU executes the pitched repetition. The size of the pitched repetition is a system parameter which can be input from the main Fortran program. The design of the real-time system assumes that the overflow of the sample buffer can be absolutely controlled by the pitched repetition. Thus, the size of the repetition has to be carefully considered.
6. The APS produces addresses for the predictor. The APU executes the predictor which employs a fourth order prediction.
7. The APU checks the LEVEL counter and sets the flag G1, if the counter is over the maximum number of samples allowed to be executed by the PARC-Transmitter.

8. The APU executes the reciprocal function to compute $1/(RMS)^2$ and $1/\sigma$. To do the reciprocal of a 32-bit floating point number, the APU usually employs an 8 x 256 read only memory which has only 8-bit precision in the mantissa part. Thus, a special subalgorithm is used to increase the precision.
9. The APS produces addresses for the pitch extraction. The APU executes the pitch extraction.
10. The APS checks the SAMPLE buffer and sets the flat G1 if the buffer is empty.
11. The APU and the APS quantize the input sample. Three flags are employed to communicate between the APU and the APS. The way to quantize an input sample is as follows:
    A. The APU signals the APS to produce the threshold address.
    B. The APU compares the magnitude of the input sample with the threshold.
    C. If the threshold is larger than the magnitude, the APU signals the APS to stop the searching and jumps to the inverse quantizer portion. Otherwise, the APU checks whether the threshold is the largest one. If it is, the APU signals the APS and jumps to inverse quantizer portion. If it is not, control is returned to Step A.

    This design allows variable quantizer levels and reduces searching time, because most of the input samples are dropped in the first two levels.
12. The APU and the APS execute the inverse quantizer.
13. The APU updates and checks the BIT counter and sets the flag AF3 if the 157-bit is reached.

14. The APU and the APS update the predictor parameters.

15. If the flag G1, which indicates an underflow of the SAMPLE buffer or reaching the maximum number of samples permitted in the time slots, is set, the APU goes to Step 16. If the flag AF3, which indicates the condition of BIT counter, is set, the APU goes to Step 17. Otherwise, the APU sets the flag AFO and goes to Step 6. The APS waits for the signals from the APU. If the flag AFO is set, it goes to Step 6. Otherwise, it goes to Step 17.

16. If this step is executed, there has been an underflow in the SAMPLE buffer, i.e., less than 157 bits are generated. The APU updates the BIT counter to account for a NULL code which indicates underflow. If the BIT counter is still less than 157 bits, the BIT counter is reset to synchronize the counters between the PARC-Transmitter and the encoder.

17. The APU and the APS output the contents of the system counters and parameters.

18. The AP interrupts the CSPU to indicate that it is finished.

12.3 The PARC Receiver

In this section, the PARC-Receiver, which resides in the AP, will be presented. The corresponding decoder and channel synchronizer will be explained in Section 12.5. The principal elements of the receiver, which is shown in Fig. 2.8, are also part of the transmitter. So, the implementation of receiver is much the same as that of the transmitter.

Before the description of the design of the PARC-Receiver, the buffer system has to be depicted. There are six buffers employed by the receiver as shown in Table 12.3.

Table 12.3 Buffer Configuration in the PARC-Receiver

Name	Bus#	Starting Location	Ending Location	Buffer Size	Buffer Type
AOM	2	3584 3840	3835 4091	252 252	Double Buffer, Short Floating Point
VHAT	3	2048	3071	1024	Circular Buffer, Short Floating Point
SHAT	2	2048	3071	1024	Circular Buffer, Short Floating Point
PARAMETER	2	4096	4255	80	Single Buffer, Long Floating Point
LEVEL	1	26000 28000	26599 28599	600 600	Double Buffer, Long Fixed Point
BIT	1	29000	30023	1024	Circular Buffer, Long Fixed Point

Among them, the AOM and the LEVEL buffers are double buffers which allow one processor to fill one-half of the buffer while another processor to process on the other half. This increases the parallel ability of the receiver. The VHAT, the SHAT and the BIT buffers are circular buffers which allow continuous access. However, the additional address pointers, which indicate the starting locations of these circular buffers at the beginning of each time slot, have to be considered. The structures of the LEVEL buffer and the BIT buffer will be explained in Section 12.5. The PARAMETER buffer is a single buffer which stores system parameters such as predictor coefficients, quantizer output scaling factors and quantizer expansion factors.

The flow chart of the receiver is shown in Fig. 12.11. The detailed functions are described as follows and the parallel processing is shown:

1. In response to the function list, the CSPU loads the APU and the APS modules of the PARC-Receiver program into the APU and the APS respectively.
2. The CSPU sets the flag RI which will initiate the APS module.
3. The APS then sets the flag RA which will start the APU module.
4. The APS produces 126 input addresses of the SHAT buffer and 252 output addresses of the AOM buffer. The APU processes the upsampling operation on 126 input reconstructed speech samples and produces 252 upsampled data points. The upsampling operation employs the linear interpolation technique which is explained in Section 2.8. The APU also limits the values of output data to be in the range of [2047/2048,-2048/2048] which is acceptable for the AOM.

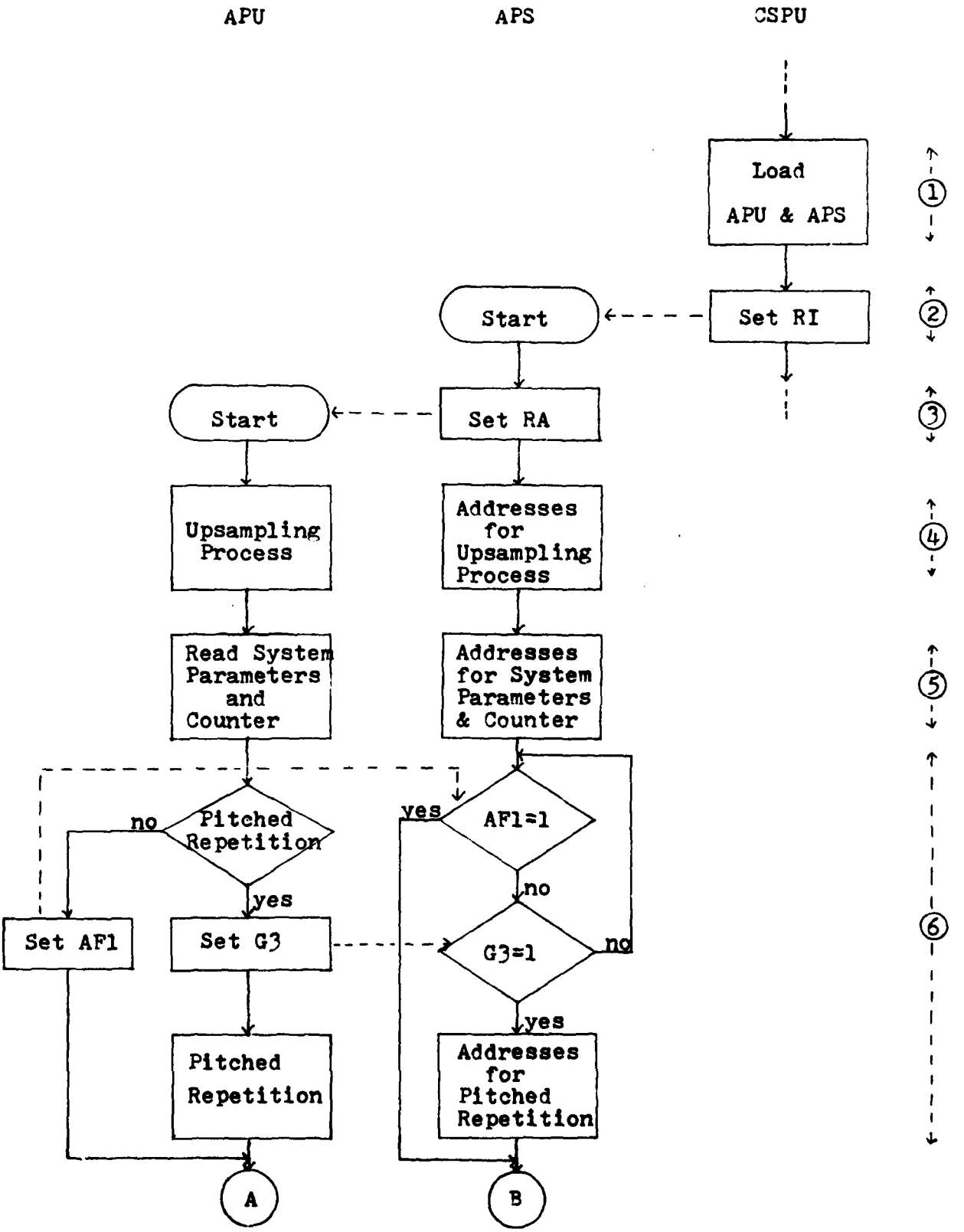


Fig. 12.11 The Flow Chart of the PARC-Receiver Program

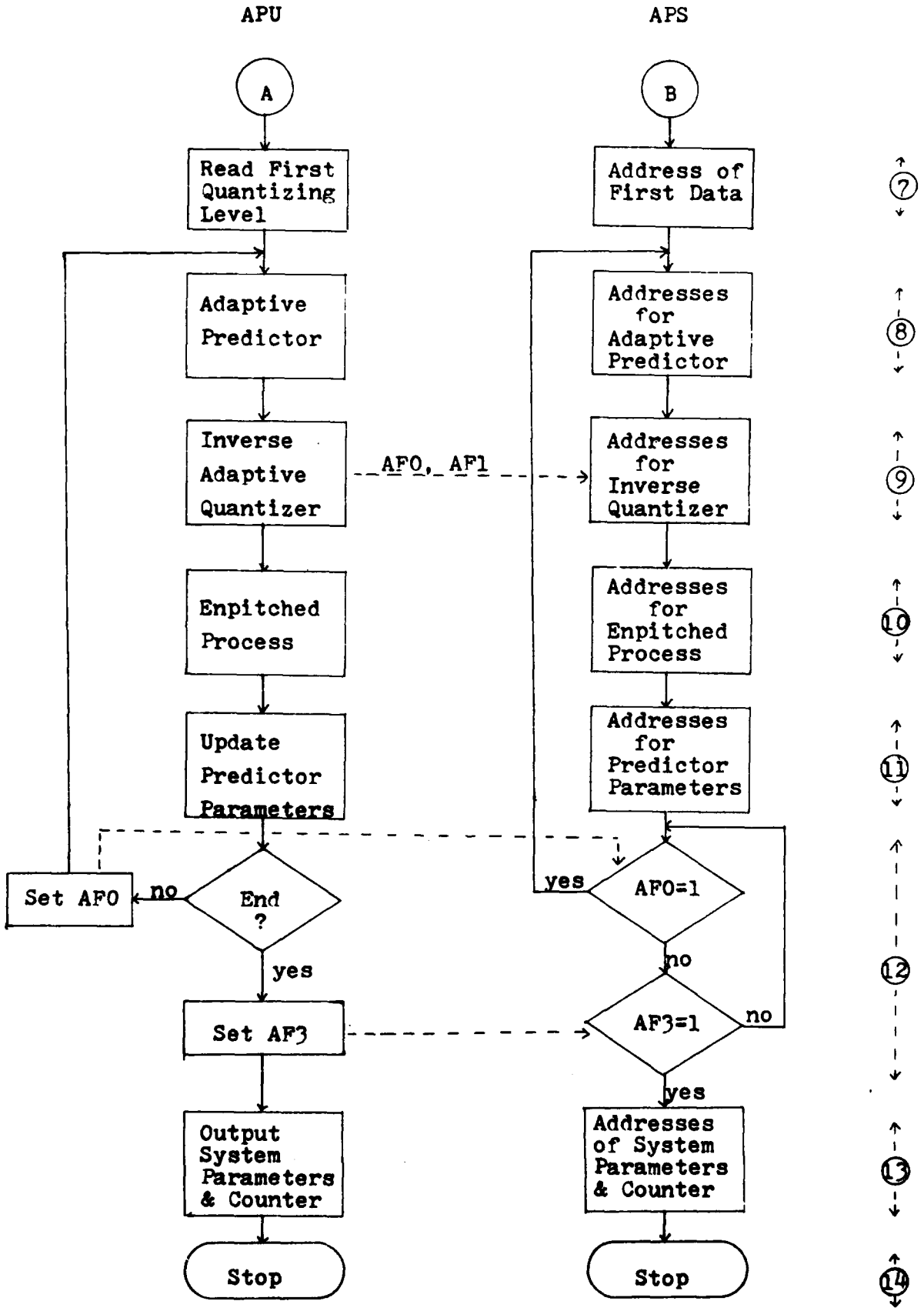


Fig. 12.11 The Flow Chart of the PARC-Receiver Program

5. The APS produces addresses of the system parameters and a counter. The APU reads the system parameters and the counter. There is only one system counter, the SAMPLE counter, which is used to count the total number of samples processed by the PARC-Receiver during a time slot. The receiver address pointer update program will use this information to update the address pointers of the receiver circular buffers.
6. The APU checks the first data in the LEVEL buffer and decides the condition of the pitched repetition. If no pitched repetition is employed, it sets the flag AF1 and goes to Step 7. Otherwise, it sets flag G3 and processes pitched repetition. The APS waits for signals from the APU. If the flag AF1 is set, it goes to Step 7. If the flag G3 is set, it produces addresses for the pitched repetition operation.
7. The APS produces the second data which is the first received quantizing level in the LEVEL buffer. The APU reads the first quantizing level.
8. The APS produces the addresses for the adaptive predictor. The APU processes the adaptive predictor which employs a fourth order prediction.
9. The AP operates the inverse adaptive quantizer. Two flags, AF0 and AF1, are used to communicate between the APU and the APS.
10. The APS produces the address of $\hat{V}(K-T)$. The APU restores the pitch redundancy as shown in Eq. (2.1).
11. The APS produces addresses of the predictor parameters. The APU updates the predictor coefficients.

12. The APU reads the next data in the LEVEL buffer and checks whether it has reached the end of the buffer. If it has, the APU sets the flag AF3 and goes to Step 13. Otherwise, it sets the flag AF0 and goes to Step 8. The APS waits for signals from the APU. If the flag AF3 is set, it goes to Step 13. If the flag AF0 is set, it goes to Step 8 to continue the receiver loop.
13. The AP outputs the system parameters and the counter to the MAP memory.
14. The AP interrupts the CSPU to indicate that it is finished with the PARC-Receiver.

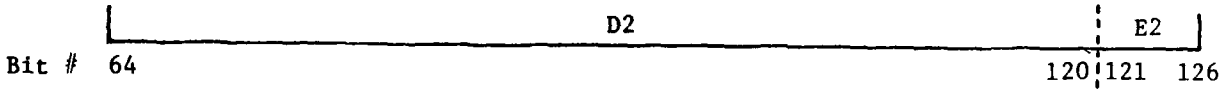
12.4 Noiseless Source Coder

This section describes the implementation of the noiseless source coder on the central processor of the MAP 300 called CSPU. As discussed in Section 2.7, the source coder performs several functions. Its primary purpose is encoding the quantizer levels and the side information for each block of N_B samples processed by the PARC transmitter. In addition, it supplies the bit pattern for frame synchronization and the parity code for channel error control.

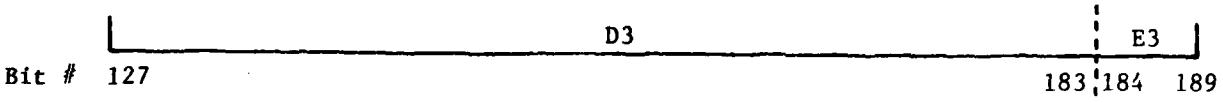
At its input, the encoder shares a double buffer containing the pitched repetition indicator and N_B quantizer levels each with the PARC transmitter. The corresponding pitch reduction parameters β,T are in two sets of locations in the scalar table. At its output, it shares a double buffer containing 189 bits each with the IOS, the processor which interfaces with a digital modem.

The encoder strives to form a frame of 189 bits from the quantizer levels and the side information generated by the transmitter. The frame is subdivided into three Hamming blocks of 63 bits each. A Hamming block contains 57 information bits and 6 parity bits. Each of the 57 information bits in a block is protected against errors. Depending on whether or not pitched repetition is indicated, the output frame assumes one of two formats Figs. 12.12 and 12.13. The encoder proceeds sequentially to build the different fields in the output frame. It handles one Hamming block at a time. The parity codeword for the block is updated each time a bit is output. Immediately after the 57th information bit in a Hamming block is transferred, the parity codeword for that block is output. The codeword is then initialized for the next block, and the encoder continues to output further information bits. This

Hamming Block 1



Hamming Block 2



Hamming Block 3

Field	Contents
A	Synchronization bit
B	Pitch period T
C	Pitch correlation coefficient β
D1, D2, D3	Quantizer levels
E1, E2, E3	Parity bits for error control

Fig. 12.12 Normal Format for Bit Frame Output of Encoder

Hamming Block 1

Field	Contents
A	Synchronization bit
B	Pitch period T
C'	False β, for pitched repetition
C	Pitch correlation coefficient β
D1	Quantizer levels
E1	Parity bits for error control

Fig. 12.13 Format of first Hamming Block during pitched repetition

process is repeated for three Hamming blocks, completing the output frame of 189 bits.

Fig. 12.14 shows the flow chart for the encoder. The program listing is contained in Appendix G. The encoder starts with field A, Fig. 12.12. It fetches the previous sync bit from a location where it was saved and inverts it. This new value is saved for use in the next frame. It is also output to field A. Next the encoded value of T is fetched and transferred to the output buffer, field B. The value of the pitched repetition indicator is then checked. If affirmative, the code 0000000 is transferred to field C′, Fig. 12.13. Next the partially encoded value of β is fetched. It must be between 0 and 96 indicating one of 97 possible levels. Using this as an index, a 7-bit code is retrieved from the β encode table and output to field C, Figs. 12.12, 12.13. After this the quantizer levels are fetched one by one. The corresponding code-lengths and code-words are retrieved from the Q-level encode table and transferred to the output buffer until the fields D1, D2, and D3 are filled. This should coincide with the exhausting of the N_B Q-levels generated by the transmitter.

An exception to this occurs when the sample buffer runs out. The block of quantizer levels do not generate enough bits to fill up the fields D1, D2, and D3. In this case, a null code 1111111 0 is output to indicate the end of quantizer level information. If there is still more space available, it is padded in with 1's. The null code and the padding bits are error-protected the same as any other information bits.

The source code used here is a variable-length to variable-length mapping, and it is not necessary that fields D1, D2, and D3 are exactly filled by encoding an integer number of quantizer levels. The extra bits after a

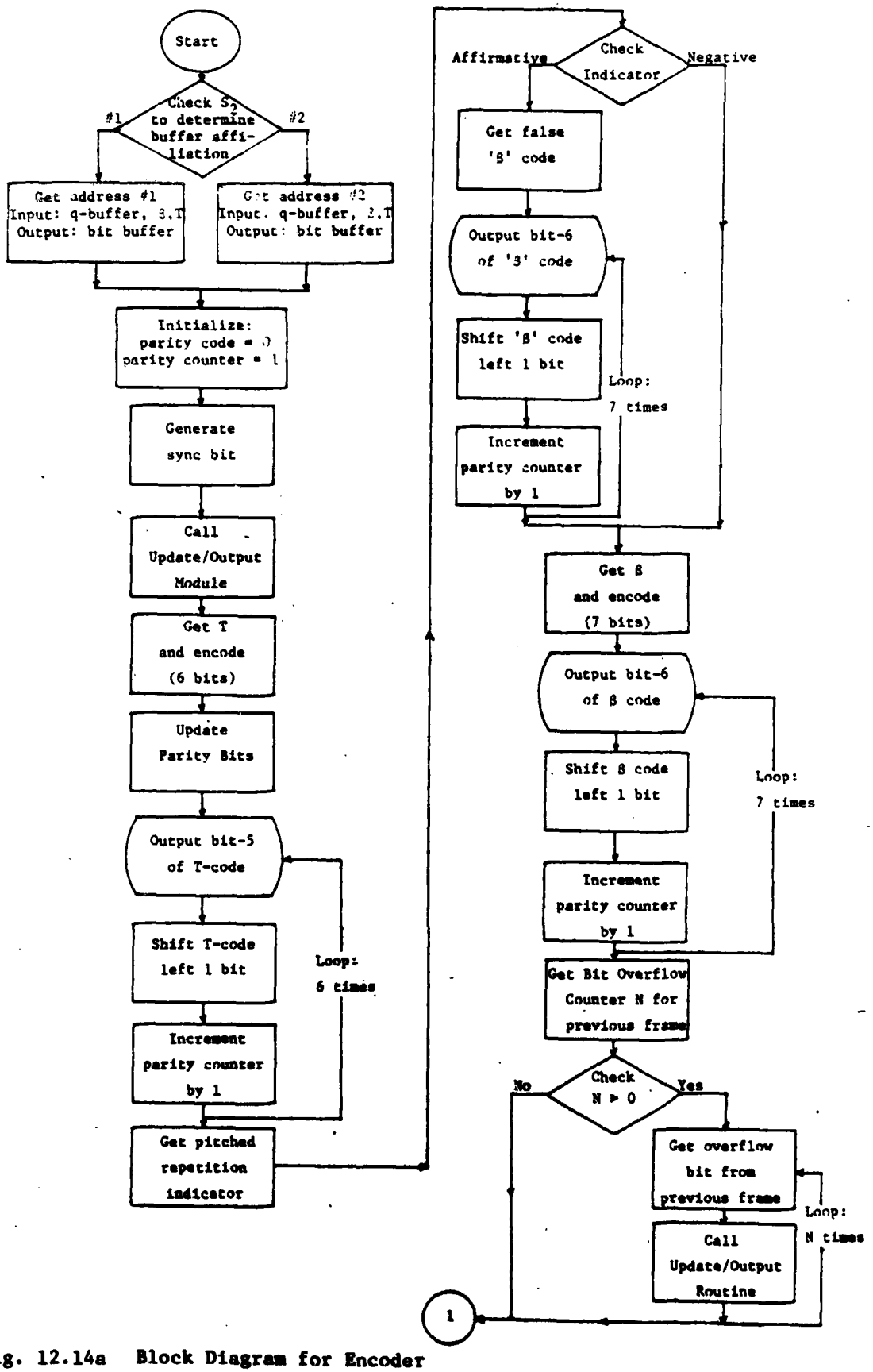


Fig. 12.14a Block Diagram for Encoder

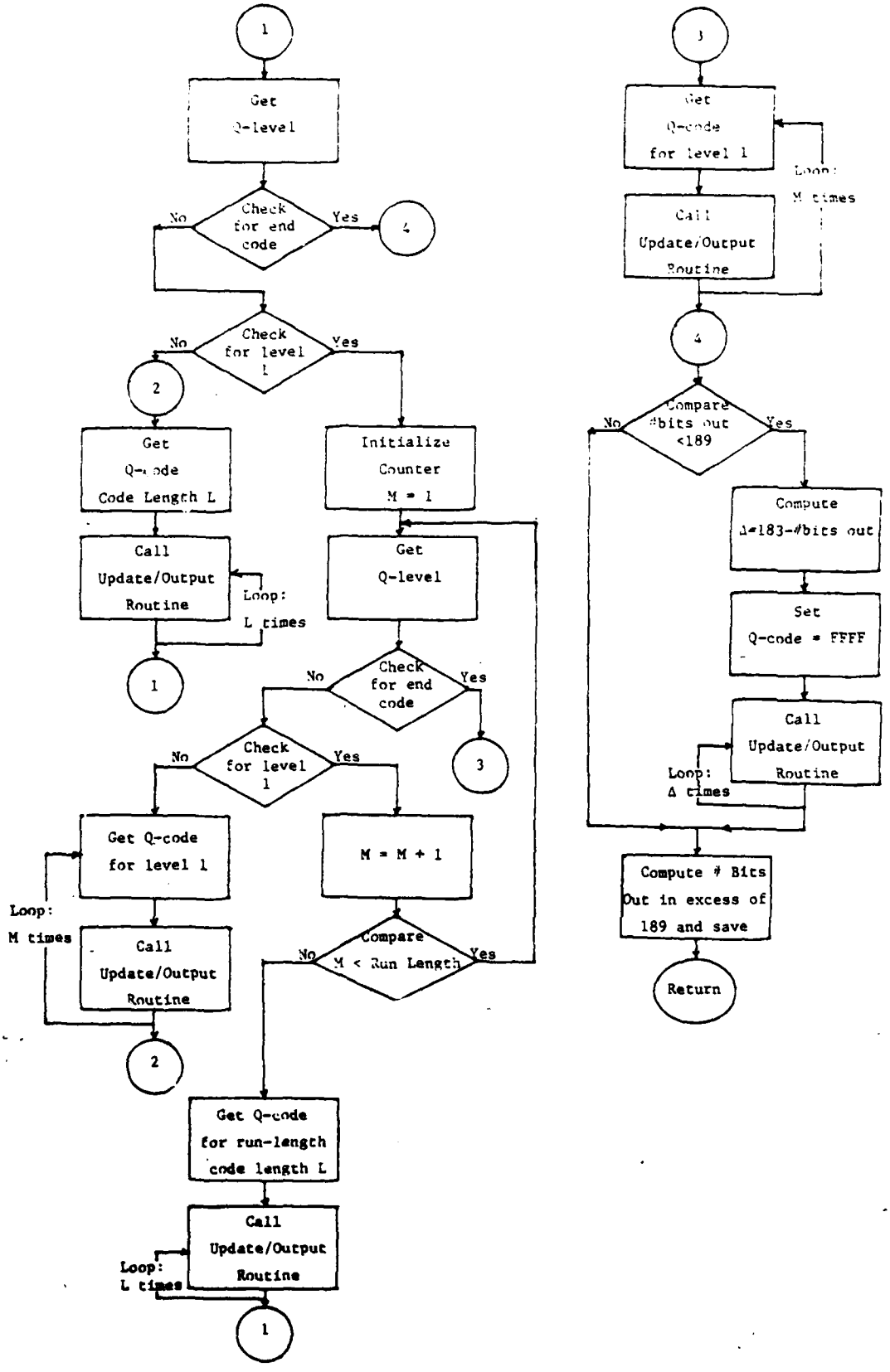
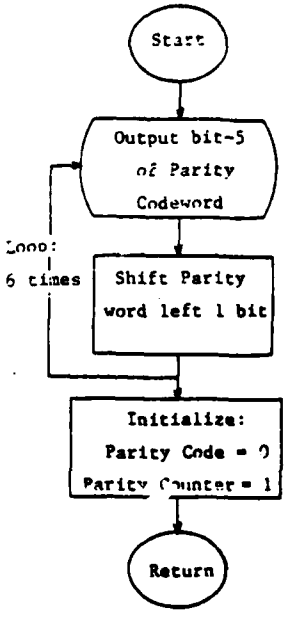


Fig. 12.14b Block Diagram for Encoder

A) Parity Output Routine



B) Update/Output Routine

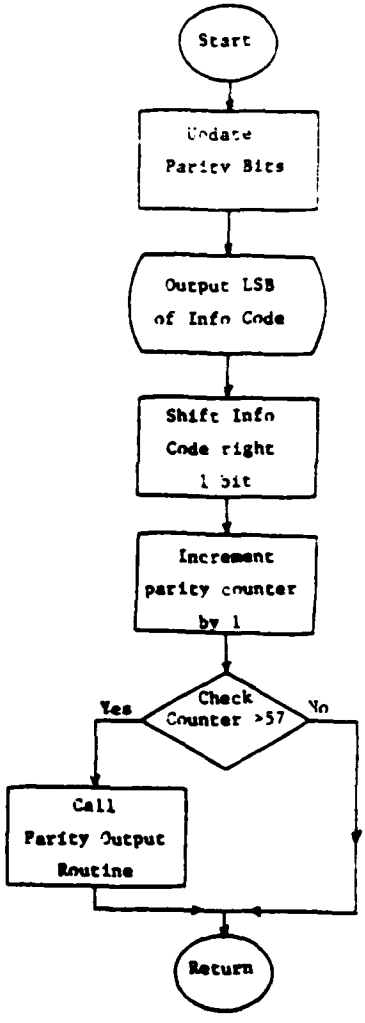


Fig. 12.14c Block Diagram for Encoder

Hamming block is full are passed to the appropriate field in the next Hamming block.

All the programs in each processor need to be able to process a block of information within a time frame of 19687.5 μ sec for the algorithm to keep in real time. This necessitated giving program efficiency priority over flexibility. The following is a list of the constraints on the encoder program as a result of this:

(i) All buffers used by the encoder must be located in Bus 1. They can be relocatable, as long as they do not occupy memory locations above 64K.

(ii) All the inputs to the encoder program must be in the fixed number format. The quantizer levels must be pre-multiplied by 2 to facilitate direct indexing. The values of β and T must be partially encoded. T must be limited to 64 values from 0 to 63, and β to 97 values from 0 to 96. Pitched repetition and the end of quantizer levels must be indicated by negative numbers.

(iii) The source code has the following constraints. The code for level 1 is 0. The locations for the run length code and the null code in the Q-level encoding table are 0 and 24, respectively. The other 11 codes for the quantizer levels Q are at locations 2Q.

(iv) Because the LSB of a code-word is transmitted first, the code-word received at the decoder is inverted, e.g., 110 → 011. To compensate for this, the code-words stored in the Q-encoding table must be inverted and right justified.

(v) The Q-encoding table must contain (L-1) for the various code lengths, L.

(vi) The code to indicate pitched repetition is 0000000. The 97

quantized values of β may not use any pattern with 2 or less 1's in it. This is to give the pitched repetition indicator additional error protection.

(vii) The run length value can be selected at initialization.

(viii) Finally, the block of N_B quantizer levels should not generate more than 196 bits. Since the encoder does not monitor the maximum number of bits generated, the transmitter must ensure this limit.

12.5 Synchronizer, Decoder

This section describes the implementation of the decoder on the central processor of the MAP 300 called CSPU. The decoder essentially inverts at the receiver the operation performed by the encoder at the transmitter. From each received frame of 189 bits, it generates a block of N_B quantizer levels, the pitched repetition indicator, and the pitch extraction parameters for use by the PARC receiver. In addition, it performs several other operations. It monitors transmission errors, and corrects up to one error per Hamming block in the received frames. Because the received bit stream is arranged in frames, the decoder acquires the initial frame synchronization using the synchronization bit pattern transmitted by the encoder. Later, it monitors the received sync pattern to ascertain that synchronization is not lost. It monitors the state of the sample buffer in the PARC receiver to make sure the buffer does not overflow or underflow. Finally, it controls the mode of operation of the entire PARC algorithm.

The different operations outlined here are performed in different modules in the decoder. The decoder program is subdivided into three modules: the initialization module; the synchronization acquisition module; and the decoder module. The function to be performed and correspondingly the module to be used is determined by the three values of a function select switch, S1. The first time the program goes in for sync acquisition, several parameters and buffers need to be initialized. This mode is indicated by a value of 0 for S1. Subsequent sync acquisition operations are indicated by a value of -1 for S1. After frame sync has been established, the decoder can perform its task of inverting the bit stream to quantizer levels. This mode is indicated by a value of 1 for S1.

The decoder shares a double buffer with the PARC receiver at its output. A second function select switch, S2, indicates which half of the double buffer the decoder must use in the current time frame.

At its input, the decoder shares a circular buffer 1024 bits long with the IOS. The decoder expects to receive 189 bits in each time frame. However the clock that controls the inflow of the bit stream is at the other end of the communication system. It is different from the clock that measures the time frame at the decoder. On the average, the decoder does receive 189 bits per time frame. However, because of the different clocks, there could be temporary deviations. The large circular buffer which is several times the frame size allows for these deviations without causing a conflict between the decoder and the IOS over the bits being read out of and written into the buffer. The IOS, which receives the bit stream from the digital channel, has its base pointer which indicates the position of the base of the current frame to be entered into the buffer. When the PARC algorithm is initiated, the base pointer of the decoder which indicates the base of the frame of bits it should process in the current time frame is set several blocks behind the input pointer controlled by the IOS. Each processor updates its pointer independently at the end of the time frames. As long as the input and output rates match on the average, no conflict should occur and the two pointers should remain reasonably separated.

On being called, the decoder checks S1 and transfers control to the appropriate module. The module performs its function, and then it sets up the control and updates the relevant parameters and information for the operation in the next time frame. The program listing of the decoder is included in Appendix G. The following subsections describe in detail the operation of the three modules that make up the decoder.

12.5.1 Synchronization Acquisition Module

Before the decoder can start inverting the received bit stream, the boundaries of the frames, marked by the sync bit, must be located. This task is achieved by the synchronization module, over several time frames. The sync acquisition algorithm is described in Chapter 3. The implementation here varies slightly from the description in Chapter 3. This is to reduce the amount of computation and memory storage required and to make the program more time efficient. The block diagram of the sync acquisition module is shown in Fig. 12.15.

The encoder inserts an oscillating bit S_i in field A, Fig. 12.12, for synchronization. The decoder generates a similar pattern of expected sync bit values, $\hat{S}_i$. During sync acquisition, the correlation of $\hat{S}_i$ and each of the 189 bit positions starting from the base pointer is computed. A cumulative correlation tally from one time frame to the next for each of the bit positions is stored separately. Just in case the sync sequence being generated at the decoder is the inverse of the sync pattern being received from the transmitter, a cumulative tally of correlations with $\bar{\hat{S}}_i$ is also saved for each of the bit positions. The bit position corresponding to the maximum cumulative tally in each time frame is kept. When the correlation tally for a particular bit is maximum for ten consecutive frames, it is assumed the sync bit has been located. That position, marking the beginnings of frames generated by the encoder, is saved.

If at the end of a sync acquisition operation, it is determined that sync has not yet been acquired, the program sets up for one more sync operation in the next time frame. The base pointer in the circular buffer is updated by 189. The current value of $\hat{S}_i$ is inverted and saved. And the function select switch S2 is changed to indicate opposite buffer affiliations

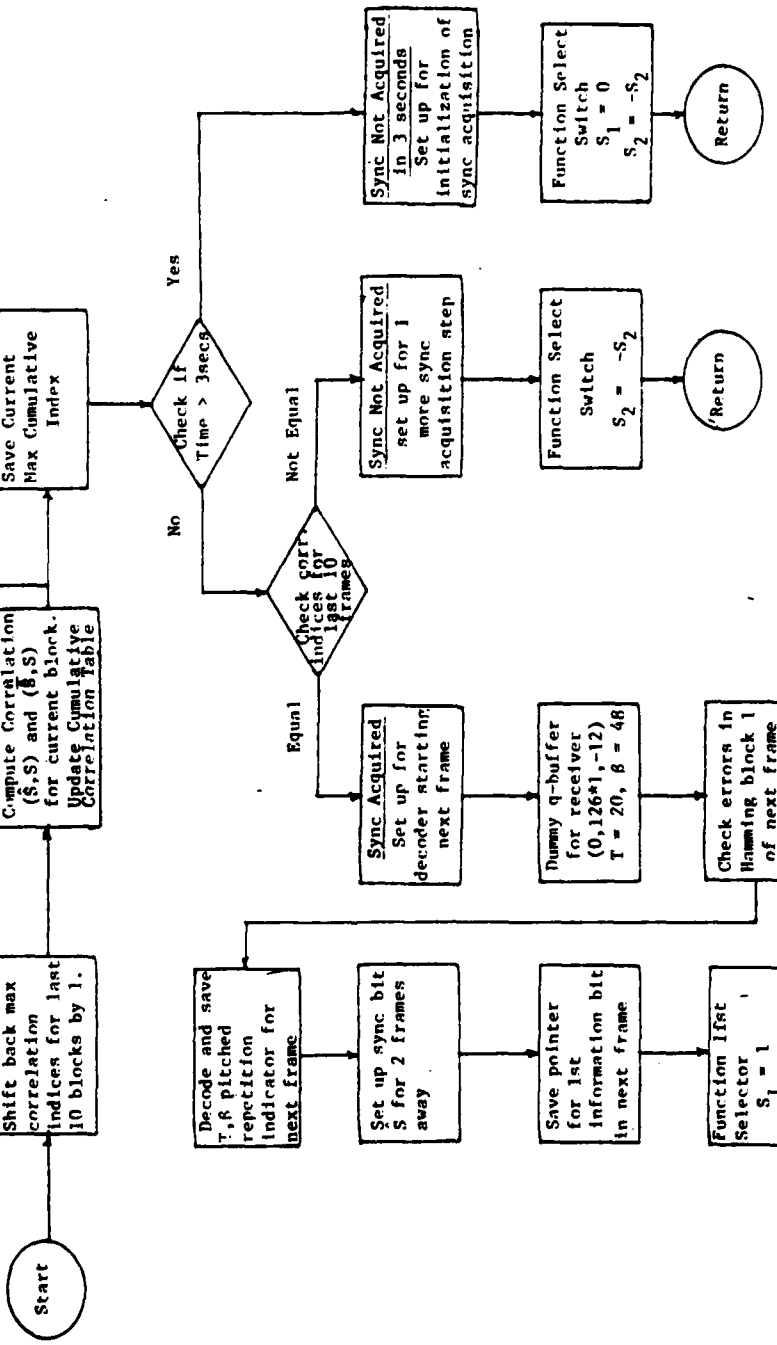


Fig. 12.15 Block Diagram for Sync Acquisition

for the next time frame.

If synchronization is achieved, preparations are made for subsequent decoder operations. In addition to the above controls and parameters, several others are updated. Function select switch S1 is changed to 1 to indicate a decoder operation in the next frame. Although no decoder operation has so far been performed, a dummy buffer consisting of 126 level 1's is prepared for use by the PARC receiver. Error check is conducted on the first Hamming block of the next bit frame; and up to one error is corrected. T, β and the pitched repetition indicator for the next frame are decoded and saved for output later. This leads to the location of the first bit in the field B1, Fig. 3.12, of the next frame. This is called the 1st information bit, information about quantizer levels. Its location is saved for use in the next frame. The cost function for sync monitor is initialized to -1. And the expected sync value $\hat{S}_{i+1}$ is set to match the next received sync value S_{i+1}. Finally, the state of the receiver sample buffer is initialized to a buffer-full condition. The control is then returned to the MAP executive.

12.5.2 Decoder Module

This module is the heart of the decoder. It corrects for transmission errors, if possible; monitors frame synchronization; inverts the received bit sequence to information usable by the PARC receiver; and performs buffer control on the receiver sample buffer. Its block diagram is shown in Fig.12.16.

After it decides its buffer affiliation by checking S2, it does error detection and correction on the 2nd and 3rd Hamming blocks of the current bit frame and the 1st Hamming block of the next bit frame. The reason for this offset between the bit frame and the error control frame is to make the extra bits generated during the encoding of the current frame available to the decoder when it performs the decoding. The extra bits reside at the

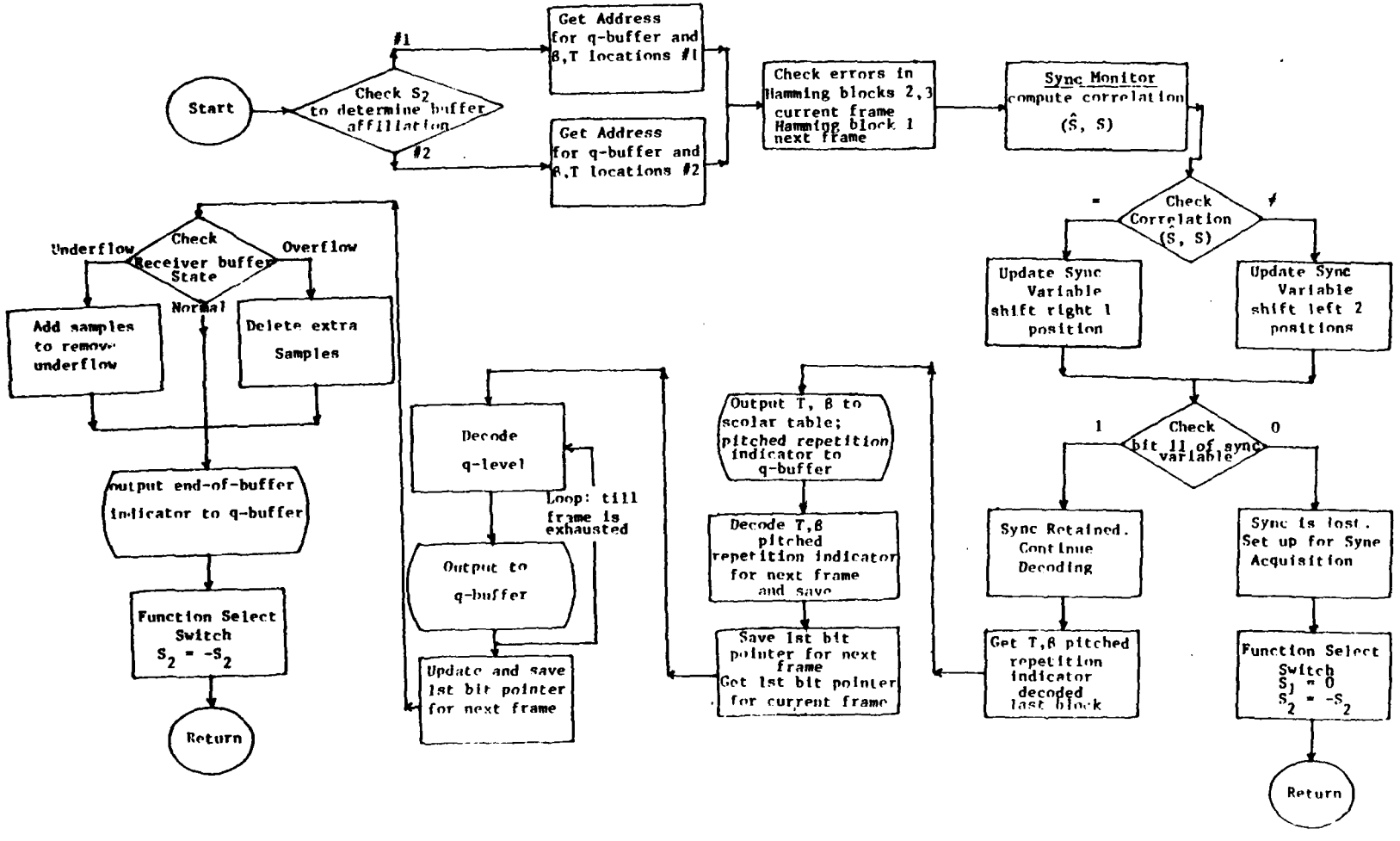


Fig. 12.16 Block Diagram for Decoder

beginning of field B1, Fig. 12.12 of the next frame.

The next step is sync monitor. The basis of the algorithm is explained in Chapter 3. The sync variable starts with an initial value of -1, FFFF in hexadecimal representation. The received sync bit S_i is compared with its expected value $\hat{S}_i$ using the logical operation EXOR. If equal, the bits in the sync variable v_i are shifted right one position. Bit 15, the sign bit is retained 1 in this operation. If not equal, the sync variable is shifted left two positions, causing 0's to be inserted in the least significant bits of v_i.

Bit 11 of v_i is monitored. If it is 0, sync is declared lost. The module sets the switch S1 to 0 to indicate a series of sync acquisition operations starting in the next time frame, and returns control to the MAP executive.

As long as bit 11 of v_i is 1, sync is declared preserved, and the module continues to its next phase of operation, source decoding. The pitch reduction parameters β and T which were decoded and saved in the previous time frame are output to their appropriate locations in the scaler table. The pitched repetition indicator, also decoded in the previous time frame, is transferred to the output buffer. If affirmative, the receiver buffer pointer is updated by the length of the repetition block.

After this, the pitch reduction parameters and the pitched repetition indicator for the next time frame are decoded and saved. This leads to the position of the 1st information bit in the next frame. If there were any extra bits in the current frame, the decoder will know where to look for them when it needs them.

The source decoding is a tree searching operation along the coding tree. Each successive bit received represents a node in the tree. Starting from the root of the tree, bits are read in, until the last bit received represents the termination of the tree. The corresponding quantizer level(s) are determined

from the decoding table and transferred to the output buffer. This process continues until either a null code is received or all the bits in the current frame are exhausted. At the end of the the decoding operation, the location of the 1st information bit for the next frame is saved.

The decoder can correct transmission errors to a point, after which errors filter through to the received bit stream. Errors have two effects on the decoding procedure. The quantizer levels generated are wrong. And the number of quantizer levels N_B in the current block can be different from what it should have been. This causes the state of the sample buffers in the transmitter and receiver to lose correspondence. To correct for this, the decoder provides a steady drift in the state of the receiver buffer. When N_B for a received block is 126, it indicates that the transmitter was processing silence when it generated this block and its sample buffer was empty. Correspondingly, the receiver buffer should be full. A few extra level 1's are added to the q-buffer generated for the receiver; so that if errors have caused the receiver to move away from its buffer-full condition, it should slowly drift back. This alleviates the mismatch between the buffer states at the receiver and the transmitter.

Finally, the state of the receiver sample buffer is checked. If it underflows, extra level 1's are added to remedy it's negative state. If it overflows, the extra samples are deleted.

After this, the module updates various controls and parameters for another decoder operation in the next time frame. It then returns control to the MAP executive.

12.5.3 Initialization Module

The initialization module precedes the first of a sequence of sync

acquisition operations. Sync acquisition requires two buffers initialized for its proper operation. This is done by the initialization module. The locations which contain the bit positions with max correlation tallies from the last ten frames are initialized to -1. The buffer where the correlation tallies are stored is initialized to 0. The switch S1 is set to -1 to indicate subsequent sync acquisition operations.

In addition, the length of the pitched repetition block is retrieved from the function control block and saved for later use in the decoder. It also gets the base address of the input buffer. It then transfers control to the sync acquisition module.

12.5.4 Decoder Constraints

As with the other modules in the PARC algorithm, the stress in developing the decoder program is execution efficiency rather than flexibility. This is to try and meet the rather tight limitations of real time operation. The following constraints are imposed on the decoder:

(i) All buffers used by the decoder must reside on Bus 1. They can be relocatable, as long as they do not occupy memory locations above 64K.

(ii) The input for this program is a circular buffer of length 1024. The information bit must be contained in the LSB of each word.

(iii) All the outputs of the decoder are in the fixed point number format. The β and T are partially encoded and are represented by 7 and 6 bits respectively. β is allowed one of 97 codewords, 0 to 96, and T one of 64, 0 to 63. Pitched repetition and end-of-quantizer-levels are indicated by negative numbers.

(iv) The run length value can be selected at initialization.

12.6 Program Timing and Speed

The programs in the real time implementation of PARC get executed in parallel on the different processors of the MAP 300. The set of programs on a given processor must be executed once in each time frame. The number of samples processed by these programs is not constant. It varies from about 50 during the voiced regions of speech to about 520 during the transition regions following voiced speech. For the algorithm to operate at its peak performance, the programs should be able to process the maximum number of 520 samples within a time frame. This is, however, limited by processor speeds and program efficiency.

The average number of samples that need to be processed by any program is 126; and it is imperative that the programs be able to process at least a few extra samples if necessary. Currently, the programs do satisfy this minimum requirement. The pitched reduction program, PARC transmitter, and PARC receiver which operate on the AP can process about 150 samples per time frame. The encoder and decoder programs which operate on the CSPU can process about 200 samples per time frame. So currently, the limit on the number of samples processed is set at 150. This exceeds the minimum requirement only marginally, and speeding up the programs would improve the performance of the algorithm considerably. Some ways to achieve this improvement are suggested below, and it is expected that different programs will execute 20% to 50% faster as a result of these modifications.

The pitched reduction program which currently requires about 5 msec can be speeded up 100% by doubling the parallelism in its computation. The transmitter and receiver programs need to be reorganized in terms of their register usage and operation sequencing. This should provide approximately 30% speed

improvement. These changes should increase the number of samples processed on the AP by about 50%.

The encoder can be speeded up 25% by rearranging the encoding table and changing all memory access from indirect to direct addressing. This would require the input quantizer levels to be 2(q-1) instead of 2q as they are now. The decoder can be improved slightly by rearranging its decoding table and operation sequencing.

Two of the three buses on the MAP 300 have slower MOS memories. Changing these to the fast bipolar memory would also help the speed on the various processors because of the large number of memory transfers performed in each time frame. With these modifications, it is expected the algorithm would be able to process 250-300 samples per time frame instead of the 150 it does now. Although this still allows the algorithm to operate well below its peak level of fidelity, this is about the level of performance that can be expected given the current speed of the MAP 300 array processor.

APPENDIX E
RESAMPLING PROGRAM

During the course of this project, because of the different sampling rates at which digitized speech was available from various sources and the different rates at which it was utilized, it was found necessary to develop an efficient resampling program to enable us to easily change sampling rates of speech. The following briefly describes the resampling algorithm and the operational details of the program. Fig. E.1 shows the flow chart, and program listings are included at the end of this appendix.

Sampled speech is the pulse amplitude representation of an analog speech signal at a sequence of time points separated by T_1. At another sampling rate, it is the PAM representation of the same signal at a different set of points separated by T_2. The new set can be obtained by interpolation from the first set of samples. So as also to limit the bandwidth of signals, interpolation was performed by discrete convolution of the original samples with the truncated impulse response of an ideal low pass filter. A first order Hamming window was used for truncation to reduce the effects of truncation. The impulse response and the Hamming window have the following form:

$$H_F(t-\tau) = 2F \text{ sinc } (2F\pi\tau) \qquad -\infty < \tau < \infty$$

and

$$W_T(t-\tau) = \alpha + (1-\alpha)\cos(\pi\tau/2T) \qquad -T < \tau < T$$

$$\alpha = 0.7$$

The convolution function is the product of these two. The discrete version of the convolution function is scaled down by T_1, to compensate for the scaling introduced by discrete filtering.

$$C_i(nT_2-(m-i)T_1) = \frac{2F}{T_1}\{\alpha + (1-\alpha)\cos[((mT_1-nT_2) + iT_1)/NT_r]\}$$
$$\{\text{sinc}[2F\pi((mT_1-nT_2) + iT_1)]\}, \qquad \frac{-NT_r}{T_1} < i < \frac{NT_r}{T_1}$$

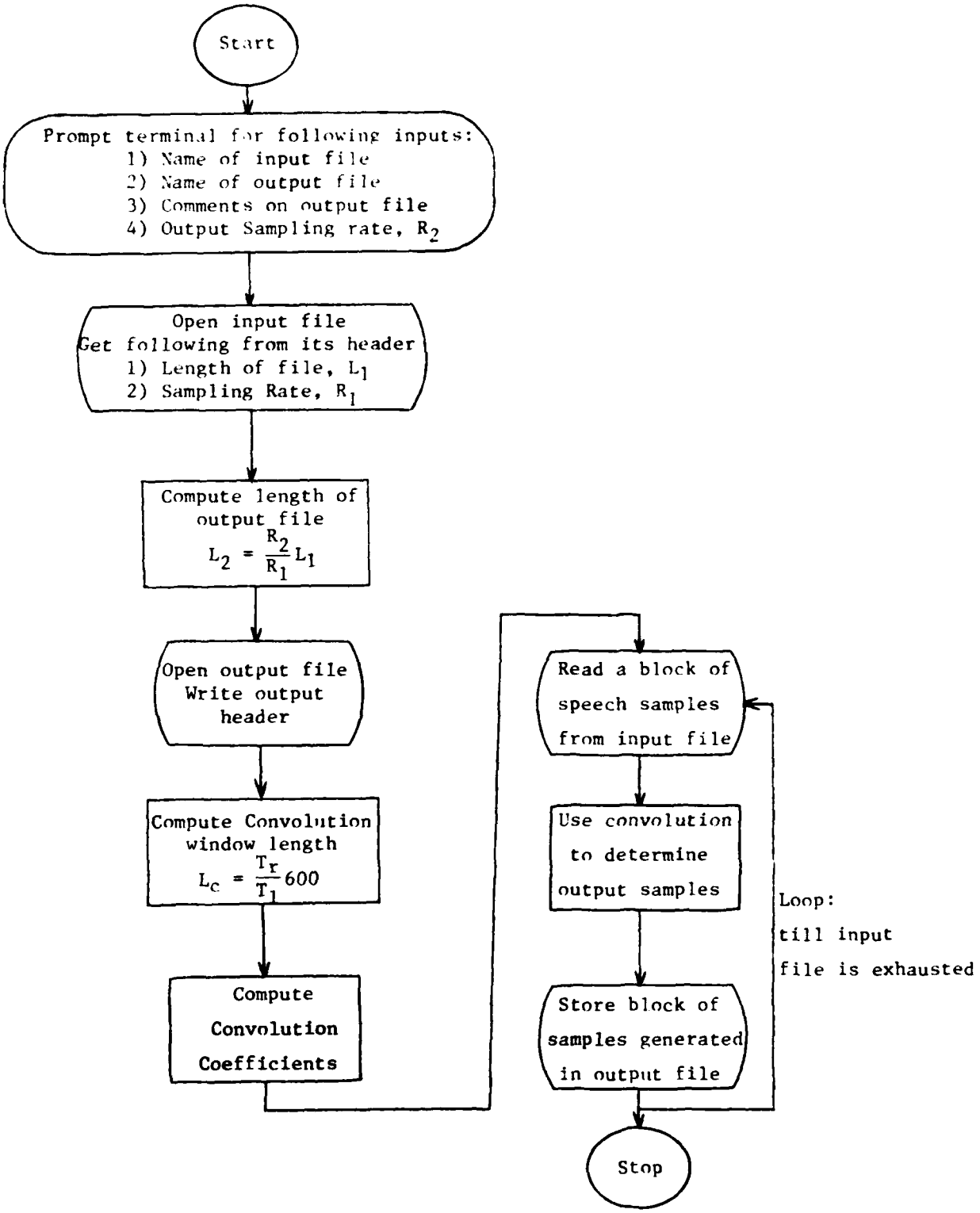


Fig. E.1 Flow Chart for the Resample Program

Here nT_2 identifies the nth output sample and mT_1 the closest preceding input sample.

Because the phase $(mT_1 - nT_2)$ changes arbitrarily, the correlation coefficients need to be computed for each output sample. However, if the two sampling rates are integer submultiples of a higher frequency $1/T_r$ called the reference frequency, the C_i need to be evaluated only once at the reference frequency. The phase $(mT_1 - nT_2)$ is always a multiple of T_r, and a subset of these coefficients can be utilized to perform the convolution. This reduces the computation required in the program by about an order of magnitude. For the different rates required in this project, the frequency 96000 is an integer multiple and was selected as the reference frequency.

Two versions of the resampling program were developed. In the first, all computation, initialization as well as the convolution (which comprises the bulk of the computation in resampling) is done on the PDP 11/60. In the second, the convolution is performed on the MAP 300 array processor using the SNAP-II software package of array functions supplied by CSPI. The PDP 11 does the initialization and handles the I/O interfacing between the disk and the MAP 300. The second version is almost an order of magnitude faster than the first.

E.1 Operating Details

The program is segmented into four modules. The main program acquires all relevant parameters required for the resampling operation, either from the header record of the input speech file or from the terminal. This program prompts for each input from the terminal, and if requested, gives a brief description, shows an acceptable example, and specifies the bounds on the value of the input. The second module performs the convolution for resampling. The other two modules assist in the initialization by computing the convolution coefficients. Between the two versions of the program only the second module is different. The appropriate module is selected during the process of task-building.

To obtain an executable task image, the appropriate modules must be compiled and task-built. The following describes this process for the RSX11M operating system on a PDP 11 computer. The compilation step for each module has the following form.

```
>FOR  MODULE = MODULE/I4
```

The task-building step is slightly different for the two versions. The command format for the first version is:

```
>TKB
TKB> QRESAM/CP/FP = MOD1, MOD2, MOD3, MOD4
TKB> /
ENTER OPTIONS:
TKB> UNITS = 7
TKB> ACTFIL = 7
TKB> //
```

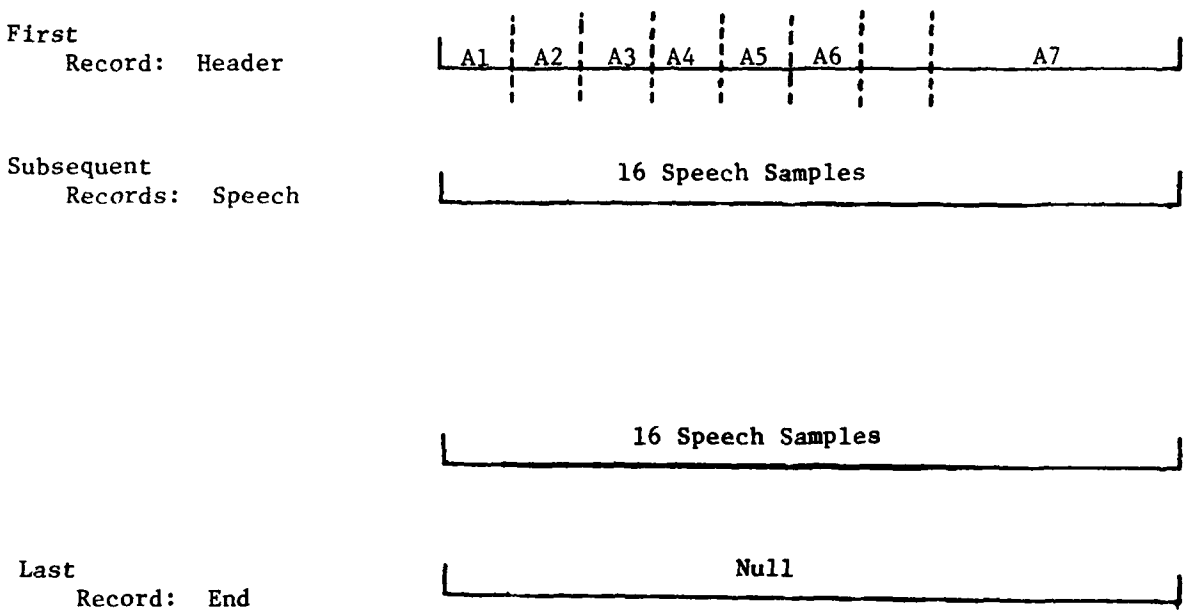
The underlined text is the prompt by the computer. The following text is the response by the user.

For the second version, the SNAP library must also be included in in the task-building step.

```
>TKB
TKB>  MRESAM/CP/MP = MOD1, MOD2, MOD3, MOD4, SNAPLIB/LB
TKB>  /
ENTER OPTIONS:
TKB>  UNITS = 10
TKB>  ACTFIL = 10
TKB>  //
```

The program requires the input speech file to conform to the Notre Dame speech file format. Fig. E.2 illustrates this format. The first record is the header. The seven fields in it are sentence identification number, sampling rate, number of samples in the files, lower and upper cutoff frequencies, truncation number for the convolution window, and comments identifying the speech file. The upper cutoff frequency is not used in this program. The truncation number which specifies the length of the convolution window is internally computed by the program based on the input sampling rate. Each of the subsequent records in the file contain 16 speech samples in the I5 format. The amplitude of speech samples is limited between -2048 and 2047. The speech file must end with a null record.

The output file generated by the resampling program also conforms to this format.



Format for each record of 80 columns

Header: (6I5, 10X, 2042)

Speech: (16I5)

Fig. E.2 Notre Dame Speech File Format

```
FORTRAN IV-PLUS V02-51          16:14:00    15-APR-80          PAGE 1
QRESAMINF.FTN   /I4/TR:BLOCKS/WR

      C
      C*****************************************************************
      C
      C                 ****************************************
      C                 *                                      *
      C                 *       MAINLINE INFO QRESAMPLE        *
      C                 *                                      *
      C                 ****************************************
      C
      C
      C                              ARVIND S ARORA
      C
      C    THIS PROGRAM SEEKS OUT THE INFORMATION NECESSARY TO
      C RUN THE QRESAMPLE SUBROUTINE. IT THEN OPENS THE INPUT AND OUTPUT
      C FILES,WRITES THE OUTPUT HEADER AND TRANSFERS CONTROL TO
      C QRESAMPLE. THE FOLLOWING INFORMATION IS SOUGHT:
      C     1)NAM1      INPUT FILE NAME
      C     2)NAM2      OUTPUT FILE NAME
      C     3)IRATE1    INPUT SAMPLING RATE (FROM INPUT FILE HEADER)
      C     4)IRATE2    OUTPUT SAMPLING RATE
      C     5)NLEN1
      C     5)NLEN1     LENGTH OF INPUT FILE (FROM INPUT FILE HEADER)
      C     6)NLEN2     LENGTH OF OUTPUT FILE, ONLY IF PARTIAL
      C                 OUTPUT IS DESIRED
      C     7)IF1       FILTER CORNER FREQ
      C                 (RESTRICTED TO INTEGER SUB-MULTIPLES OF 9600)
      C     8)IF2       (IGNORE)
      C     9)ITRUN     TRUNCATION NO. FOR FILTER
      C                 NO. OF TERMS IN FILTER=2*ITRUN+1
      C                 (THIS IS DETERMINED BY THE PROGRAM DEPENDING ON IF1)
      C    10)HEAD1     INPUT FILE HEADER COMMENTS (FROM INPUT FILE)
      C    11)HEAD2     OUTPUT FILE HEADER COMMENTS
      C
      C
      C*****************************************************************
      C
0001            INTEGER*2 NAM1(16),NAM2(16),HEAD1(20),HEAD2(20)
0002            LOGICAL*1 ANS,CHAR(3)
0003            COMMON /RESAM/NLEN1,NLEN2,IRATE2
0004            COMMON /COFF/IF1,IF2,ITRUN,IRATE1
0005            DATA CHAR/' ','N','Y'/
0006            TYPE *,'    SOLICITING INFORMATION FOR RESAMPLING'
0007            TYPE *,' '
0008            TYPE *,' '
0009            TYPE 1009    !WANT TO SEE EXAMPLES?
0010            ACCEPT 1003,ANS
0011            IFL=1
0012            IF(ANS.EQ.CHAR(3)) IFL=2
      C ----- BEGIN QUERY
0013      100   CONTINUE
      C ----- NAME OF INPUT FILE
0014            IF(IFL.EQ.1) GO TO 110
0015            TYPE *,'** EXAMPLE OF FILE NAME **'
0016            TYPE *,'    DK:[50,50]S11564.DAT;1'
0017      110   TYPE 1010    !ENTER INPUT FILE NAME
0018            ACCEPT 1001,NAM1
0019            NAM1(16)=0
```

```
        C ----- OPEN INPUT FILE. IF ERROR IN OPENING, INDICATE ERROR
        C       AND ASK AGAIN
#02#            OPEN(UNIT=1,NAME=NAM1,TYPE='OLD',READONLY,SHARED,ERR=12#)
#021            READ(1,1###) ISEN,IRATE1,NLEN1,IF1,IF2,ITRUN,HEAD1
        C ----- ECHO INPUT FILE NAME AND INPUT HEADER
#022            TYPE 1##4,NAM1
#023            TYPE 1##5,ISEN,IRATE1,NLEN1,IF1,IF2,ITRUN,HEAD1
#024            GO TO 13#
#025    12#     CONTINUE
#026            TYPE *,'** ERROR -- FAILED TO OPEN SPECIFIED FILE'
#027            TYPE *,'    POSSIBLE REASONS: FILE DOES NOT EXIST'
#028            TYPE *,'                      DEVICE NOT AVAILABLE'
#029            GO TO 11#
#03#    13#     CONTINUE
        C ----- OUTPUT FILE PARAMETERS
#031            TYPE *,'** RESAMPLING PARAMETERS'
        C ----- OUTPUT SAMPLING RATE
#032            IF(IFL.EQ.1) GO TO 14#
#033            TYPE *,'** EXAMPLE OF OUTPUT SAMPLING RATE **'
#034            TYPE *,'    THE USUAL RATES FOR SPEECH ARE 6###<RATE<16###'
#035            TYPE *,'    FOR EXAMMPLE 64##'
#036    14#     CONTINUE
#037            TYPE 1#11    !OUTPUT SAMPLING RATE
#038            ACCEPT *,IRATE2
#039            TYPE *,IRATE2
        C ----- CHECK IF RATE IS LEGAL
#04#            IF(IRATE2.GT.#.AND.96###/IRATE2*IRATE2.EQ.96###) GO TO 15#
#041            TYPE *,'** ERROR -- CAN''T HAVE RATE<#'
#042            TYPE *,'          OR SUBMULTIPLE OF 96###'
#043            GO TO 14#
#044    15#     CONTINUE
        C ----- CORNER FREQUENCIES
#045            IF(IFL.EQ.1) GO TO 16#
#046            TYPE *,'** EXAMPLE OF FILTER CORNER FREQUENCIES F1 **'
#047            TYPE *,'    THE USUAL RANGE IS  RATE/2>F1>#'
#048            TYPE *,'    FOR EXAMPLE    3###'
#049    16#     CONTINUE
#05#            TYPE 1#12     !CORNER FREQ
#051            ACCEPT *,IF1
#052            TYPE *,IF1
        C ----- CHECK IF FREQS. ARE LEGAL
#053            IF(IF1.GT.#) GO TO 17#  !LEGAL
#054            TYPE *,'** ERROR -- ILLEGAL FREQUENCY'
#055            TYPE *,'    F1>#'
#056            GO TO 16#
#057    17#     CONTINUE
        C ----- TRUNCATION NUMBER
#058    18#     CONTINUE
#059            ITRUN=6##*IRATE1/96###
#06#            TYPE *,ITRUN
        C ----- DETERMINE LENGTH OF OUTPUT FILE
#061            NLEN2=FLOAT(NLEN1)*FLOAT(IRATE2)/FLOAT(IRATE1)
#062            NBLK=NLEN2/1##+1
#063            TYPE *,'** THE ESTIMATED LENGTH OF THE OUTPUT FILE IS'
#064            TYPE *,'    ',NLEN2,' SAMPLES, OR ',NBLK,' BLOCKS ON DISK'
#065            TYPE 1#13    !ASK IF PARTIAL OUTPUT WANTED
```

```
0066            ACCEPT 1003,ANS
0067            IF(ANS.NE.CHAR(3)) GO TO 200  !COMPLETE OUTPUT
0068            TYPE 1014   !DESIRED LENGTH OF FILE
0069            ACCEPT *,NLEN2
0070            TYPE *,NLEN2
0071    200     CONTINUE
        C ----- OUTPUT FILE NAME
0072            TYPE 1019  !PROMPT FOR OUTPUT FILE NAME
0073            ACCEPT 1001,NAM2
0074            TYPE 1004,NAM2
0075            NAM2(16)=0
        C ----- HEADER COMMENTS
0076            TYPE *,'** ENTER HEADER COMMENTS, 40 CHARACTERS OR LESS'
0077            IF(IFL.EQ.1) GO TO 210
0078            TYPE 1015, HEAD1
0079    210     CONTINUE
0080            TYPE 1016    !HEADER COMMENTS PROMPT
0081            ACCEPT 1001,HEAD2
0082            TYPE *,'** THE HEADER CARD FOR THE OUTPUT FILE IS:'
0083            TYPE 1005,ISEN,IRATE2,NLEN2,IF1,IF2,ITRUN,HEAD2
        C ----- ONE OPPORTUNITY TO CHANGE ENTRIES
0084            TYPE 1017    !WANNA CHANGE SOMETHING
0085            ACCEPT 1003,ANS
0086            IF(ANS.EQ.CHAR(3)) GO TO 100  !DO WANT TO CHANGE
0087            OPEN(UNIT=2,NAME=NAM2,TYPE='NEW',CARRIAGECONTROL='LIST',ERR=220)
0088            WRITE(2,1000) ISEN,IRATE2,NLEN2,IF1,IF2,ITRUN,HEAD2
0089            TYPE *,'** BEGINNING RESAMPLING. WILL INFORM YOU ',
                1'ON COMPLETION'
0090            CALL RESAMP(1,2)
0091            STOP
0092    220     CONTINUE
0093            TYPE *,'** ERROR -- FAILED TO OPEN OUTPUT FILE'
0094            TYPE *,'   LOCATE PROBLEM AND TRY AGAIN'
0095            STOP
0096    1000    FORMAT(6I5,10X,20A2)
0097    1001    FORMAT(20A2)
0098    1002    FORMAT(6I5)
0099    1003    FORMAT(A1)
0100    1004    FORMAT(1X,20A2)
0101    1005    FORMAT(1X,6I5,10X,20A2)
0102    1009    FORMAT('$** DO YOU WISH TO SEE EXAMPLES OF RESPONSES? (Y,N):')
0103    1010    FORMAT('$** ENTER INPUT FILE NAME. NAME=')
0104    1011    FORMAT('$** ENTER RESAMPLING RATE. RATE=')
0105    1012    FORMAT('$** ENTER CORNER FREQ  F1=')
0106    1013    FORMAT('$** WOULD YOU LIKE A PARTIAL OUTPUT? (Y,N):')
0107    1014    FORMAT('$** ENTER LENGTH DESIRED. LENGTH=')
0108    1015    FORMAT(' ** EXAMPLE:',20A2)
0109    1016    FORMAT('$** COMMENTS:')
0110    1017    FORMAT('$** DO YOU WISH TO CHANGE ENTRIES? (Y,N)')
0111    1018    FORMAT('$** ENTER TRUNCATION NUMBER. ITRUN=')
0112    1019    FORMAT('$** ENTER OUTPUT FILE NAME. NAME=')
0113            END
```

```
C
C*********************************************************************
C
C       ****************************************
C       *                                      *
C       *       SUBROUTINE  RESAMP(NIN,NOUT)   *
C       *                                      *
C       ****************************************
C
C                         ARVIND S ARORA
C
C PROGRAM DESCRIPTION:
C
C    THIS PROGRAM RESAMPLES A GIVEN FILE AT ONE SAMPLING
C RATE TO THAT AT ANOTHER SAMPLING RATE. BECAUSE THE COEFFS
C REQUIRED FOR THE FILTERING OPERATION DURING RESAMPLING ARE
C COMPUTED ONLY ONCE AT 96### SAMPLES/SEC, THE INPUT AND
C OUTOUT FREQUENCIES ARE RESTRICTED TO INTEGER SUB-MULTIPLES
C OF 96###, E.G. 16###, 12###, 8###, 6###. HOWEVER THIS MAKES
C THIS PROGRAM OPERATE AN ORDER OF MAGNITUDE FASTER THAN THE
C GENERAL VERSION OF RESAMP. ALL COMPUTATIONS IN THIS PROGRAM
C ARE DONE IN THE PDP-11 CPU.
C    THE PROGRAM REQUIRES THE FOLLOWING INFORMATION BEFORE IT
C CAN RESAMPLE A GIVEN FILE:
C    1) NIN     LUN FOR INPUT FILE
C    2) NOUT    LUN FOR OUTPUT FILE
C    3) NLEN1   LENGTH OF INPUT FILE
C    4) NLEN2   LENGTH OFOUTPUT FILE
C    5) IRATE1  SAMPLING RATE OF INPUT FILE
C    6) IRATE2  SAMPLING RATE OF OUTPUT FILE
C    7) IF1     CORNER FREQ OF FILTER
C    8) IF2     (IGNORED)
C    9) ITRUN   (IGNORED)
C
C    IN ADDITION THE INPUT AND OUTPUT FILES MUST BE OPEN
C FOR ACCESS.
C
C
C DATA FORMATS:
C
C
C    BOTH THE INPUT & OUTPUT SAMPLES MUST HAVE VALUES V IN THE
C RANGE -2#48 TO 2#47 (12 BIT RESOLUTION). THEY MUST BE STORED
C USING A (16I5) FORMAT.
C
C
C SIGNIFICANT VARIABLES:
C
C
C    1) A1      INPUT FILE BUFFER (INTEGER*2)
C    2) A2      OUTPUT FILE BUFFER (INTEGER*2)
C    3) H       FILTER COEFFS (REAL)
C               SYMMETRIC, CENTRE COEFF IN H(1),
C               COEFFS ONLY FOR +VE TIME
C
C
```

```
        C CALLS TO SUBROUTINES:
        C
        C
        C     1)SUBROUTINE ERRSET(24,,,,,.FALSE.,)
        C     2)FUNCTION-SUBROUTINE SECNDS(TIME)
        C     3)SUBROUTINE COEFF(H)?)
        C
        C
        C*********************************************************************
        C
0001            SUBROUTINE RESAMP(NIN,NOUT)
0002            INTEGER*2 A1(320),A2(160)
0003            DIMENSION H(601)
0004            COMMON /RESAM/NLEN1,NLEN2,IRATE2
0005            COMMON /COFF/IF1,IF2,ITRUN,IRATE1
0006    100     CONTINUE
0007            NLEN2=NLEN2-1
0008            IF(9600/IRATE1*IRATE1.NE.9600) GO TO 110
0009            IF(9600/IRATE2*IRATE2.NE.9600) GO TO 110
0010            GO TO 120
        C ----- INDICATE ILLEGAL FREQUENCIES AND STOP
0011    110     CONTINUE
0012            CLOSE(UNIT=NIN,DISPOSE='SAVE')
0013            CLOSE(UNIT=NOUT,DISPOSE='DELETE')
0014            TYPE *,'** ERROR QUICK RESAMPLE'
0015            TYPE *,' '
0016            TYPE *,'THIS SUBROUTINE HANDLES ONLY A LIMITED SET OF FREQS.'
0017            TYPE *,'INPUT AND OUTPUT FREQS MUST BE SUBMULTIPLES OF 9600'
0018            TYPE *,'AND YOUR FREQS ARE, IN FREQ=',IRATE1,' OUT FREQ=',IRATE2
0019            STOP
0020    120     CONTINUE
0021            CALL ERRSET(24,,,,,.FALSE.,)
        C ----- INITIALIZE CLOCK TO COMPUTE TIME REQUIRED FOR RESAMPLING
0022            CLCK=SECNDS(0.0)
0023    200     CONTINUE
        C ----- INITIALIZE ALL VARIABLES  AND THE INPUT BUFFER
0024            INTV1=9600/IRATE1
0025            INTV2=9600/IRATE2
0026            NBUF1=320
0027            NBUF2=160
0028            IHOLD=ITRUN
0029            ITRUN=600/INTV1
0030            NFLG1=ITRUN+1
0031            NFLG2=NBUF1-ITRUN
0032            IBUF1=NFLG1+MOD((NBUF1-NFLG1+1),16)
0033            DO 210 IA1=1,IBUF1-1
0034            A1(IA1)=0
0035    210     CONTINUE
0036            READ(NIN,1000)(A1(IA1),IA1=IBUF1,NBUF1)
        C ----- INITIALIZE INDICES
0037            IA1=0
0038            IA2=0
0039            IBUF2=1
        C ----- CAL. SUBROUTINE TO COMPUTE COEFFICIENTS
0040            CALL COEFF(H)
0041            INTVN=INTV1+1
```

RESAMP.FTN /I4/TR:BLOCKS/WR

```
      C ----- BEGIN CONVOLUTION LOOP FOR FILTERING
#042  220     CONTINUE
      C ----- COMPUTE PHASE AND INITIALIZE INDICES FOR CONVOLUTION
#043          IPHASE=IA2*INTV2-IA1*INTV1
#044          IIN=IBUF1-1
#045          IIP=IBUF1+1
#046          IHN=INTVN+IPHASE
#047          IHP=INTVN-IPHASE
      C ----- PERFORM CONVOLUTION
#048          SUM=H(1+IPHASE)*A1(IBUF1)
#049          DO 250 III=IHN,600,INTV1
#050          SUM=SUM+H(III)*A1(IIN)+H(IHP)*A1(IIP)
#051          IIN=IIN-1
#052          IIP=IIP+1
#053          IHP=IHP+INTV1
#054  250     CONTINUE
#055          ITEMP=SUM
#056          IF(ITEMP.GE.20048) ITEMP=20047
#057          IF(ITEMP.LT.-20048) ITEMP=-20048
#058          A2(IBUF2)=ITEMP
      C ----- CONVOLUTION COMPLETED
      C
      C ----- INCREMENT VARIOUS INDICES
#059          IA2=IA2+1
#060          IBUF2=IBUF2+1
      C ----- FIND CORRESPONDING INDICES IN THE INPUT FILE
#061          INEXT=INTV2*IA2/INTV1
#062          IBUF1=IBUF1+INEXT-IA1
#063          IA1=INEXT
      C ----- CHECK IF OUTPUT FILE IS COMPLETE
#064          IF(IA2.LE.NLEN2) GO TO 300  !IF NOT COMPLETE, PROCEED
      C ----- IF FILE COMPLETE, WRITE OUTPUT BUFFER, CLOSE FILES & QUIT
#065          WRITE(NOUT,1000) (A2(J),J=1,IBUF2-1)
#066          NLEN2=NLEN2+1
#067          ITRUN=IHOLD
      C ----- FIGURE OUT TIME TAKEN, SAY OPERATION COMPLETE & QUIT
#068          CLCK=SECNDS(CLCK)
#069          TYPE *,' '
#070          TYPE *,'** TIME FOR RESAMPLING = ',CLCK,' SECONDS'
#071          TYPE *,' '
#072          TYPE *,'** RESAMPLING OPERATION COMPLETE.'
#073          RETURN
      C ----- CHECK IF OUTPUT BUFFER IS FULL
#074  300     CONTINUE
#075          IF(IBUF2.LE.NBUF2) GO TO 310  !IF NOT FULL, PROCEED
      C ----- IF FULL, WRITE OUT OUTPUT BUFFER & INITIALIZE FLAG IBUF2
#076          WRITE(NOUT,1000) A2
#077          IBUF2=1
      C ----- CHECK IF RUN OUT OF INPUT BUFFER
#078  310     CONTINUE
#079          IF(IBUF1.LE.NFLG2) GO TO 350  !IF NOT RUN OUT, PROCEED
      C ----- IF RUN OUT, REPLENISH BUFFER BY MOVING OLD DATA OVER
#080          IFLNXT=NFLG1+MOD((IBUF1-NFLG1),16)
#081          ISHFT=IBUF1-IFLNXT
#082          IBUF1=IFLNXT
#083          DO 320 J=ISHFT+1,NBUF1
```

```
0084            A1(J-ISHFT)=A1(J)
0085     320    CONTINUE
         C ----- FILL IN THE REST OF THE BUFFER WITH NEW DATA
0086            READ(NIN,1000,ERR=330) (A1(J),J=NBUF1-ISHFT+1,NBUF1)
0087            GO TO 350  !READ IN BUFFER, SO PROCEED
         C ----- RAN OUT OF INPUT SAMPLES SO FILL IN BUFFER WITH ZEROS
0088     330    CONTINUE
0089            DO 340 J1=J,NBUF1
0090            A1(J1)=0
0091     340    CONTINUE
         C ----- ALL SYSTEMS GO ! BACK FOR NEXT SAMPLE
0092     350    GO TO 220
         C ----- ALL THOSE LOUSY FORMAT STATEMENTS
         C
0093     1000   FORMAT(16I5)
0094            END
```

```
C
C*****************************************************************
C
C       ****************************************
C       *                                      *
C       *       SUBROUTINE  RESAMP(NIN,NOUT)   *
C       *                                      *
C       ****************************************
C
C                       ARVIND S ARORA
C
C PROGRAM DESCRIPTION:
C
C    THIS PROGRAM RESAMPLES A GIVEN FILE AT ONE SAMPLING
C RATE TO THAT AT ANOTHER SAMPLING RATE. BECAUSE THE COEFFS
C REQUIRED FOR THE FILTERING OPERATION DURING RESAMPLING ARE
C COMPUTED ONLY ONCE AT 96000 SAMPLES/SEC, THE INPUT AND
C OUTOUT FREQUENCIES ARE RESTRICTED TO INTEGER SUB-MULTIPLES
C OF 96000, AND MUST LIE BETWEEN 6400 AND 32000, FOR EXAMPLE,
C 6400, 8000, 9600, 12000, 16000, 19200, 24000, & 32000.
C    THE COEFFICIENTS REQUIRED IN THE FILTERING OPERATION
C ARE COMPUTED IN THE PDP-11 CPU. ALL THE REST OF THE
C COMPUTATIONS ARE DONE IN THE MAP-300 ARRAY PROCESSOR. ONLY
C SNAP-II SUBROUTINES ARE USED. THE INPUT AND OUTPUT BUFFERS
C ARE MULTIPLES OF 16 . THIS IS THE RECORD LENGTH USED FOR
C STORING SPEECH FILES AT NOTRE DAME UNIVERSITY FOR THE DCA
C 9600 BPS PROJECT. THIS MINIMIZES THE MANIPULATION REQUIRED
C IN THE HOST. (FOR THIS PROGRAM TO OPERATE, IT IS IMPERATIVE
C THAT SPEECH BE STORED IN THIS FORMAT, (16I5).)
C    IN THIS PROGRAM, SINGLE BUFFERS ARE USED IN THE HOST.
C THE OUTPUT TO MAP USES A SINGLE BUFFER, BUT THE INPUT
C FROM MAP USES A DOUBLE BUFFER IN MAP.
C    THE PROGRAM REQUIRES THE FOLLOWING INFORMATION BEFORE IT
C CAN RESAMPLE A GIVEN FILE:
C    1) NIN     LUN FOR INPUT FILE
C    2) NOUT    LUN FOR OUTPUT FILE
C    3) NLEN1   LENGTH OF INPUT FILE
C    4) NLEN2   LENGTH OFOUTPUT FILE
C    5) IRATE1  SAMPLING RATE OF INPUT FILE
C    6) IRATE2  SAMPLING RATE OF OUTPUT FILE
C    7) IF1     CORNER FREQ OF FILTER
C    8) IF2     (IGNORED)
C    9) ITRUN   (IGNORED)
C
C    IN ADDITION THE INPUT AND OUTPUT FILES MUST BE OPEN
C FOR ACCESS.
C
C
C DATA FORMATS:
C
C
C    BOTH THE INPUT & OUTPUT SAMPLES MUST HAVE VALUES V IN THE
C RANGE -2048 TO 2047 (12 BIT RESOLUTION). THEY MUST BE STORED
C USING A (16I5) FORMAT.
C
C
```

```
C SIGNIFICANT VARIABLES:
C
C
C     1) A1       INPUT FILE BUFFER (INTEGER*2)
C     2) A2       OUTPUT FILE BUFFER (INTEGER*2)
C     3) H        FILTER COEFFS (REAL)
C                 SYMMETRIC, CENTRE COEFF IN H(1),
C                 COEFFS ONLY FOR +VE TIME
C
C
C CALLS TO SUBROUTINES:
C
C
C     1)SUBROUTINE ERRSET(24,,,,,.FALSE.,)
C     2)FUNCTION-SUBROUTINE SECNDS(TIME)
C     3)SUBROUTINE COEFF(H)?)
C     4)SUBROUTINES IN SNAP LIBRARY
C
C
C*****************************************************************
C
0001            SUBROUTINE RESAMP(NIN,NOUT)
0002            INTEGER*2 A1(640),A2(640)
0003            INTEGER SP2047,SN2048,SFLAG
0004            DIMENSION H(601)
0005            COMMON /RESAM/NLEN1,NLEN2,IRATE2
0006            COMMON /COFF/IF1,IF2,ITRUN,IRATE1
0007    100     CONTINUE
C ----- NAMES OF VARIOUS BUFFERS IN MAP AND HOST
0008            NHR1=1   ! INPUT BUFFER IN HOST: A1
0009            NHR2=2   ! OUTPUT BUFFER IN HOST: A2
0010            MBR1=20
0011            MBR2=21
0012            MBI2=22
0013            MBR3=23
0014            MBR4=24
0015            MBR5=25
0016            MBR6=26
0017            MBI3=27
0018            MZERO=29
0019            MOR1=11
0020            MOI1=12
0021            MOR2=13
0022            MOI2=14
0023            MC0=30
0024            MC1=31
0025            MC2=32
C ----- NAMES OF VARIOUS SCALARS IN MAP
0027            SP2047=50
0027            SN2048=51
0028            SFLAG=52
0029            IF(960000/IRATE1*IRATE1.NE.960000) GO TO 110
0030            IF(960000/IRATE2*IRATE2.NE.960000) GO TO 110
0031            IF(IRATE1.GT.320000.OR.IRATE1.LT.6400) GO TO 110
0032            IF(IRATE2.GT.320000.OR.IRATE2.LT.6400) GO TO 110
0033            GO TO 120
```

```
      C ----- INDICATE ILLEGAL FREQUENCIES AND STOP
0034  11#     CONTINUE
0035          CLOSE(UNIT=NIN,DISPOSE='SAVE')
0036          CLOSE(UNIT=NOUT,DISPOSE='DELETE')
0037          TYPE *,'** ERROR QUICK RESAMPLE'
0038          TYPE *,' '
0039          TYPE *,'   THIS SUBROUTINE HANDLES ONLY A LIMITED SET OF FREQS.'
0040          TYPE *,'INPUT AND OUTPUT FREQS MUST BE SUBMULTIPLES OF 96###'
0041          TYPE *,'AND MUST LIE BETWEEN 64## AND 32###.'
0042          TYPE *,'AND YOUR FREQS ARE, IN FREQ=',IRATE1,' OUT FREQ=',IRATE2
0043          STOP
0044  12#     CONTINUE
0045          CALL ERRSET(24,,,,,FALSE.,)
      C ----- INITIALIZE CLOCK TO COMPUTE TIME REQUIRED FOR RESAMPLING
0046          CLCK=SECNDS(#.#)
      C ----- CALL SUBROUTINE TO COMPUTE COEFFICIENTS
0047          CALL COEFF(H)
      C ----- CONVOLUTION LOOP STARTING #2##
0048  2##     CONTINUE
0049          INTV1=96###/IRATE1
0050          INTV2=96###/IRATE2
0051          NPACK1=192#/INTV1
0052          NPACK2=192#/INTV2
      C ----- OPEN MAP
0053          CALL REPORT(1#,MPOPN(#))
      C ----- CONFIGURE INPUT AND CONVOLUTION BUFFERS
0054          CALL REPORT(2#,MPCLB(MBR1,#.,312#.,#,1,#))
0055          CALL REPORT(3#,MPCLB(MBR2,5#4#.,FLOAT(NPACK1),#,INTV1,#))
0056          CALL REPORT(4#,MPCLB(MBI2,5#4#.,FLOAT(NPACK1),2,2*INTV1,#))
0057          CALL REPORT(5#,MPCLB(MBR3,12##.,FLOAT(NPACK1),#,INTV1,#))
0058          CALL REPORT(55,MPCLB(MBI3,12##.,FLOAT(NPACK1),2,2*INTV1,#))
0059          SIZE=6##/INTV1+1
0060          CALL REPORT(6#,MPCLB(MBR4,12##.,SIZE,#,-INTV1,#))
0061          CALL REPORT(7#,MPEQB(MBR5,MBR2))
0062          CALL REPORT(8#,MPCLB(MBR6,5#4#.,SIZE,#,-INTV1,#))
      C ----- CONFIGURE OUTPUT BUFFERS
0063          CALL REPORT(9#,MPCLB(29,9###.,FLOAT(NPACK2),#,1,#))
0064          CALL REPORT(1##,MPEQB(MOR1,29))
0065          CALL REPORT(11#,MPCLB(29,11###.,FLOAT(NPACK2),#,1,#))
0066          CALL REPORT(12#,MPEQB(MOR2,29))
0067          CALL REPORT(13#,MPCLB(29,9###.,FLOAT(NPACK2),2,2,#))
0068          CALL REPORT(14#,MPEQB(MOI1,29))
0069          CALL REPORT(15#,MPCLB(29,11###.,FLOAT(NPACK2),2,2,#))
0070          CALL REPORT(16#,MPEQB(MOI2,29))
      C ----- CONFIGURE COEFF BUFFERS
0071          CALL REPORT(17#,MPCLB(MC#,#.,12#1.,#,1,#))
0072          CALL REPORT(18#,MPCLB(MC1,12##.,6#1.,#,1,#))
0073          CALL REPORT(19#,MPCLB(MC2,12##.,6#1.,#,-1,#))
      C ----- CONFIGURE HOST RESIDENT BUFFERS
0074          CALL REPORT(2##,MPDHB(NHR1,A1(1),A1(NPACK1)))
0075          CALL REPORT(2#2,MPDHB(NHR2,A2(1),A2(NPACK2)))
      C ----- INITIALIZE INPUT BUFFER AREA TO ZERO
0076          CALL REPORT(2#5,MPCLB(28,#.,45##.,#,1,#))
0077          CALL REPORT(2#6,MPDCV(MZERO,#,45##,#))
0078          CALL REPORT(2#7,VSMA1(28,#,MZERO,#))
      C ----- WRITE IN THE SCALARS
```

```
0070            CALL REPORT(211,MPWST(SP2#47,2#47.,1,1))
0070            CALL REPORT(212,MPWST(SN2#48,-2#48.,1,1))
0001            CALL REPORT(213,MPWST(SFLAG,1#.,1,1))
      C ----- COPY COEFFS INTO COEFF BUFFER
0002            CALL REPORT(22#,MPWDB(MC1,H(1),4,1,H(6#1)))
0003            CALL REPORT(23#,MPWDB(MC2,H(1),4,1,H(6#1)))
      C
      C ----- MAP FUNCTION LISTS
      C
      C ----- FUNCTION LIST 1  (USES OUTPUT BUFFER 1)
0004            CALL REPORT(24#,MPBFL(1))
0005            CALL REPORT(25#,VFLT(MBR2,39,MBI2,#))
0006            CALL REPORT(26#,DCVM(MOR1,INTV2,MBR1,MC#))
0007            CALL REPORT(27#,VSMA1(MBR4,1,MBR6,#))
0008            CALL REPORT(28#,VSMA1(MBR3,1,MBR5,#))
0009            CALL REPORT(29#,VCLIP(MOR1,SP2#47,MOR1,SN2#48))
0010            CALL REPORT(3##,VFIX(MOI1,38,MOR1,11))
0011            CALL REPORT(31#,MPBFO(MBI2,NHR1,2,1))
0012            CALL REPORT(32#,MPBFI(MOI1,NHR2,2,1))
0013            CALL REPORT(33#,MPEFL(1))
      C ----- FUNCTION LIST 2  (USES OUTPUT BUFFER 2)
0014            CALL REPORT(34#,MPBFL(2))
0015            CALL REPORT(35#,VFLT(MBR2,39,MBI2,#))
0016            CALL REPORT(36#,DCVM(MOR2,INTV2,MBR1,MC#))
0017            CALL REPORT(37#,VSMA1(MBR4,1,MBR6,#))
0018            CALL REPORT(38#,VSMA1(MBR3,1,MBR5,#))
0019            CALL REPORT(39#,VCLIP(MOR2,SP2#47,MOR2,SN2#48))
0100            CALL REPORT(4##,VFIX(MOI2,38,MOR2,11))
0101            CALL REPORT(41#,MPBFO(MBI2,NHR1,2,1))
0102            CALL REPORT(42#,MPBFI(MOI2,NHR2,2,1))
0103            CALL REPORT(43#,MPEFL(2))
      C ----- FUNCTION LIST 3  (MASTER FUNCTION LIST)
0104            CALL REPORT(44#,MPBFL(3))
0105            CALL REPORT(45#,MPXFL(1))
0106            CALL REPORT(46#,MPXFL(2))
0107            CALL REPORT(47#,MPEFL(3))
      C ----- END OF FUNCTION LISTS IN MAP
      C
      C ----- PREPARING FOR LOOP IN HOST
      C
      C ----- INITIALIZE VARIOUS POSITION FLAGS, INITIALIZE INPUT BUFFER
      C       BUFFER IN MAP, TURN OVER OUTPUT BUFFER TO MAP
0108            READ(NIN,1###)(A1(IA1),IA1=1,NPACK1)
0109            CALL REPORT(5##,MPWDB(MBI3,A1(1),2,1,A1(NPACK1)))
0110            CALL REPORT(51#,VFLT(MBR3,39,MBI3,#))
      C
0111            READ(NIN,1###)(A1(IA1),IA1=1,NPACK1)
0112            CALL REPORT(52#,MPWDB(MBI2,A1(1),2,1,A1(NPACK1)))
      C
0113            CALL REPORT(53#,MPFBA(NHR2))
      C
0114            IOUT=#
0115            CALL REPORT(54#,MPWHL(SFLAG,2,3))
      C ----- HOST LOOP
0116    4##     CONTINUE
      C ----- INPUT TO MAP
```

```
0117              CALL REPORT(550,MPWBA(NHR1))
        C
0118              READ(NIN,1000,ERR=450)(A1(IA1),IA1=1,NPACK1)
0119              GO TO 500
        C ----- RUN OUT OF INPUT, FILL IN WITH ZEROS
0120    450       CONTINUE
0121              DO 460 JA1=IA1,NPACK1
0122              A1(JA1)=0
0123    460       CONTINUE
        C ----- IF NOT RUN OUT, CARRY ON WITH SAMPLES READ IN
0124    500       CONTINUE
0125              CALL REPORT(560,MPFBA(NHR1))
        C ----- OUTPUT FROM MAP
0126              CALL REPORT(570,MPWBA(NHR2))
0127              IOUT=IOUT+NPACK2
0128              IF(IOUT.GE.NLEN2) GO TO 550
0129              WRITE(NOUT,1000)(A2(IA2),IA2=1,NPACK2)
0130              CALL REPORT(590,MPFBA(NHR2))
0131              GO TO 400
        C ------ END OF HOST LOOP
        C
        C ----- IF YOU GOT ALL OUTPUT SAMPLES, STOP MAP LOOP AND RETURN
        C
0132    550       CONTINUE
        C
0133              ILST2=NPACK2+NLEN2-IOUT
0134              WRITE(NOUT,1000)(A2(IA2),IA2=1,ILST2)
        C ----- FIGURE OUT TIME TAKEN, SAY OPERATION COMPLETE & QUIT
0135              CLCK=SECNDS(CLCK)
0136              TYPE *,' '
0137              TYPE *,'** TIME FOR RESAMPLING =',CLCK,' SECONDS'
0138              TYPE *,' '
0139              TYPE *,'** RESAMPLING OPERATION COMPLETE.'
        C ----- CLOSE MAP
0140              CALL REPORT(610,MPCLS(0))
0141              RETURN
        C
        C ----- AND THEN THERE'S THE FORMAT STATEMENTS
0142    1000      FORMAT(16I5)
0143              END
```

```
C
C ----- SUBROUTINE TO REPORT MAP ERRORS
C
      SUBROUTINE REPORT(I,J)
      INTEGER*2 I,J
      IF(J.EQ.0) RETURN
      TYPE *,'** MAP ERROR',J,' AT',I
      STOP
      END
```

[redacted]

```
COEFF.FTN      /I4/TR:BLOCKS/WR
C
C*****************************************************************
C
C                 ************************************
C                 *                                  *
C                 *       SUBROUTINE  COEFF(H)       *
C                 *                                  *
C                 ************************************
C
C
C                                 ARVIND S ARORA
C
C SUBROUTINE DESCRIPTION:
C
C
C     THIS SUBROUTINE COMPUTES COEFFICIENTS FOR A LOW PASS
C FILTERING OPERATION. IT COMPUTES THE COEFFICIENTS SAMPLED AT
C 96 KSAM/SEC ONCE, AND THESE CAN BE USED FOR THE ENTIRE
C RESAMPLING OPERATION. THIS SPEEDS UP THE OPERATION BY
C ALMOST ONE ORDER OF MAGNITUDE. HOWEVER, IT RESTRICTS THE
C INPUT AND OUTPUT FREQUENCIES TO INTEGER SUB-MULTIPLES OF
C 96 K, E.G. 16###, 12###, 8###, 64##.
C     THE FOLLOWING INPUTS ARE REQUIRED:
C     1) H        FILTER COEFF ARRAY   CENTRE COEFF AT H(1)
C                 SYMMETRIC, COEFFS FOR ONLY +VE TIME
C     2) IF1      CORNER FREQ FOR THE FILTER
C     3) IF2      (IGNORED)
C     4) IRATE1   SAMPLING RATE OF THE INPUT FILE
C     5) ITRUN    FILTER TRUNCATION NO. (IGNORED)
C
C     THIS FILTER USES THE IMPULSE RESPONSE (NON CAUSAL) OF A LOW-
C PASS FILTER TRUNCATED USING A 1ST ORDER HAMMING WINDOW WITH THE
C ONE COEFFICIENT ALFA=#.7.
C
C
C CALLS TO SUBROUTINES:
C
C
C     1) FUNCTION-SUBROUTINE SINC(ANGLE)
C
C
C*****************************************************************
C
#001            SUBROUTINE COEFF(H)
#002            DIMENSION H(6#1)
#003            COMMON /COFF/IF1,IF2,ITRUN,IRATE1
#004            DATA ALFA/#.7/
      C ----- COMMENCE COMPUTATION
#005            T=1./96###.
#006            PI=2.*ASIN(1.)
#007            HCEN=2*IF1*T
#008            TSPAN=12##.*HCEN
#009            H(1)=2.*FLOAT(IF1)/FLOAT(IRATE1)
#010            DO 15# I=1,6##
#011            ARG=HCEN*PI*FLOAT(I)
#012            H(I+1)=H(I)*SINC(ARG)*(ALFA+(1.-ALFA)*COS(ARG/TSPAN))
#013     15#    CONTINUE
```

```
0014            RETURN
0015            END
```

```
C
C*********************************************************************
C
C                 ******************************************
C                 *                                        *
C                 *       FUNCTION-SUBROUTINE SINC(ARG)    *
C                 *                                        *
C                 ******************************************
C
C
C                                 ARVIND S ARORA
C
C  SUBROUTINE DESCRIPTION:
C
C
C     THIS SUBROUTINE USES AN 8TH ORDER POLYNOMIAL APPROXIMATION
C  TO COMPUTE THE SINC FOR AN ARGUMENT SPECIFIED IN RADIANS.
C        SINC(ARG)=SIN(ARG)/ARG
C
C
C*********************************************************************
C
0001            FUNCTION SINC(ARG)
0002            DIMENSION C(5)
0003            DATA C/1.570796,-0.64596371,0.07968968,-0.004673766,
                1 0.0001514842/
0004            PI=ASIN(1.)
0005            X=ARG/PI
0006            IF(X.GT.2.) X=AMOD((X-2.),4.)-2.
0007            IF(X.LT.-2.) X=-AMOD((-X-2.),4.)+2.
0008            IF(X.GT.1.) X=2.-X
0009            IF(X.LT.-1.) X=-2.-X
0010            SINC=C(1)
0011            Y=1.
0012            DO 100 I=1,4
0013            Y=Y*X*X
0014            SINC=SINC+C(I+1)*Y
0015    100     CONTINUE
0016            SINALG=SINC*X
0017            SINC=SINC/PI
0018            IF(ARG.GT.PI/2.0.OR.ARG.LT.-PI/2.0) SINC=SINALG/ARG
0019            RETURN
0020            END
```

APPENDIX F CVSD ALGORITHM

In this appendix, the CVSD algorithm will be discussed. Before the design of this real-time CVSD system can be presented, it is essential to understand the structure of CVSD systems.

F.1 The CVSD System

In this section, the structure of a CVSD system is presented. The system described here is identical to that of [1]. A schematic of an encoder and decoder pair for CVSD is shown in Figure F.1.

A CVSD encoder is simply a delta modulator with an adaptive stepsize. In order to minimize the difference between adjacent samples, the input signal s(k) is normally oversampled. Typically, the sampling rate is 16 k samples per second. Since one bit per sample is transmitted, the transmission rate is 16 K bits per second.

The encoder quantizes the difference e(k) between an input speech sample and its estimate which is predicted by a first order predictor.

$$e(k) = s(k) - \alpha * \hat{s}(k-1)$$

where $\hat{s}(k-1)$ is the reconstructed speech of the input speech sample at time k-1. Then the encoder outputs a single bit, b(k), for each corresponding input sample where

$$b(k) = \begin{cases} \text{if } e(k) \geq 0 \\ \text{if } e(k) < 0 \end{cases}$$

The adaptive stepsize logic derives its output $\Delta(k)$ from the bit stream b(k) and an appropriate initial state,

$$\Delta(k) = \beta * \Delta(k-1) + g(k)$$

where

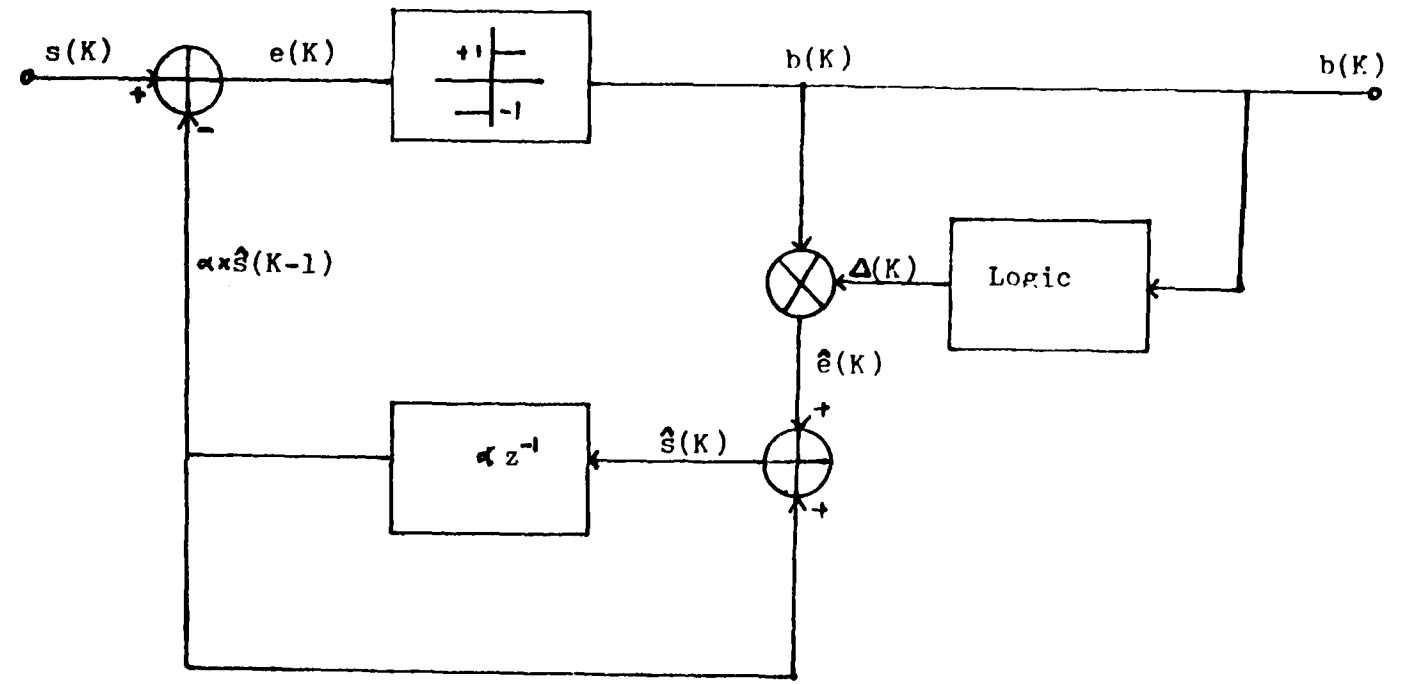


Fig. F.1 The CVSD System (a) Encoder (b) Decoder

$$g(k) = \begin{cases} \Delta_{max} * (1 - \beta) & \text{If } b(k) = b(k-1) = b(k-2) \\ \Delta_{min} * (1 - \beta) & \text{otherwise} \end{cases}$$

For a long run in which no three consecutive bits are identical, the stepsize approaches Δ_{min}. On the other hand, if the many consecutive bits are the same, the stepsize will approach Δ_{max}.

The estimate of the difference signal is given by

$$\hat{e}(k) = b(k) * \Delta(k)$$

Then the reconstructed speech is computed from

$$\hat{s}(k) = \alpha * \hat{s}(k-1) + \hat{e}(k)$$

Because the reconstructed speech in the transmitter is determined solely by the bit stream b(k) and an agreed initial state, the receiver can produce the same reconstructed speech samples if no channel error occur.

Normally, there is no best criteria to judge the performance of a speech algorithm. However, the basic signal to noise ratio criteria is used here.

$$SNR = 10 * \log_{10} \frac{\sum_{k=1}^{M} s(k)^2}{\sum_{k=1}^{M} (s(k) - \hat{s}(k))^2}$$

F.2 The Real-Time CVSD System

In this section, the design of the real-time CVSD system is presented. The major design decisions and real-time algorithm are discussed. The program listings are followed.

F.2.1 Design Overview

In order to design a real-time speech system, several points have to be considered. First, the flexibility of changing system parameters such as

speech algorithm parameters, buffer sizes and buffer starting locations must be provided. This allows system performance to be studied without any further modification of the real-time program. However, this approach will increase overhead time. Therefore, a "pre-bound" concept will be introduced. Second, the arithmetic execution time has to be minimized. This can be done by efficiently utilizing the arithmetic processors. Therefore, the "hiding" technique which hides additions inside the period of multiplication will be employed. Third, in order to decrease overhead time, all processors have to be utilized as asynchronously and as simultaneously as possible. However, the necessary synchronization between the processors has to be considered.

The real-time CVSD implementation employs five MAP processors to do speech processing asynchronously and simultaneously. The whole algorithm is divided into two steps: An initialization step and a processing step. Each processor has its own role in each step.

1. The Initialization Step

   In this step, the host computer will initialize the MAP-300 system, set up the appropriate command sequence, and initiate the real-time CVSD algorithm. Each processor in the MAP-300 has to do the following functions:

   A. The CSPU acts as the resource controller. Responding to a host command, it loads a data acquisition program into the ADAM, loads digital-to analog conversion program into the AOM , configures the buffers, loads an arithmetic function into the AP which will reset the containts of the buffers, builds a recurrent sequence of commands called a function list and prebinds the APS module of the CVSD program.

B. The AP executes the arithmetic function which is loaded by the CSPU. The purpose of this execution is to reset the buffers.

2. The Processing Step

After initialization, the MAP-300 can execute its functions without any further commands from host computer. The role of each processor are described as follows:

A. The CSPU acts as the resource controller. Responding to each interrupt, it updates the containts of the corresponding processor control block. Responding to the function list, it checks the availability of the appropriate buffer and initiates AP operation by loading the CVSD program from MAP memory. The synchronization is also done by CSPU which is described in the next section.

B. The APU executes the real-time CVSD algorithm.

C. The APS produces data addresses and acts as the controller of the real-time CVSD system.

D. The ADAM samples the input speech signal from telephone module and stores them into MAP memory.

E. AOM converts input digital samples into an analog signal and output it to telephone module.

In the next two subsections, the detailed description of these two steps will be presented.

F.2.2 The Initialization Step

The initialization step, which is controlled by host computer, includes reading system parameters, giving commands to the CSPU and building a function list. The algorithm is shown in Fig. F.2.

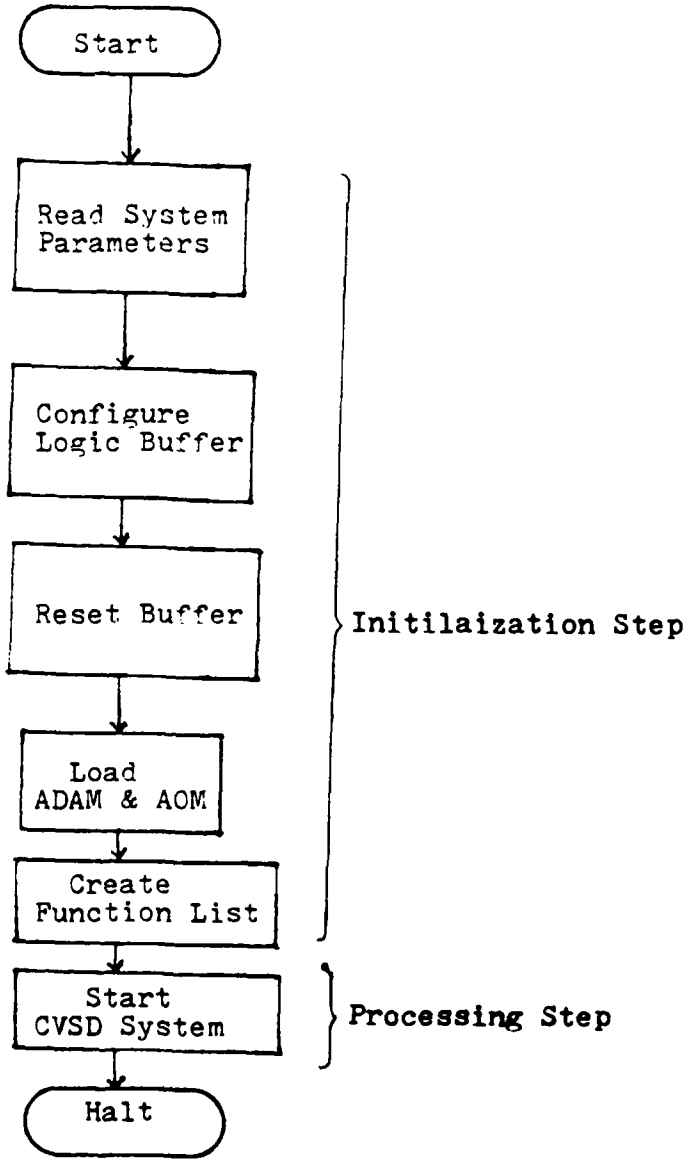
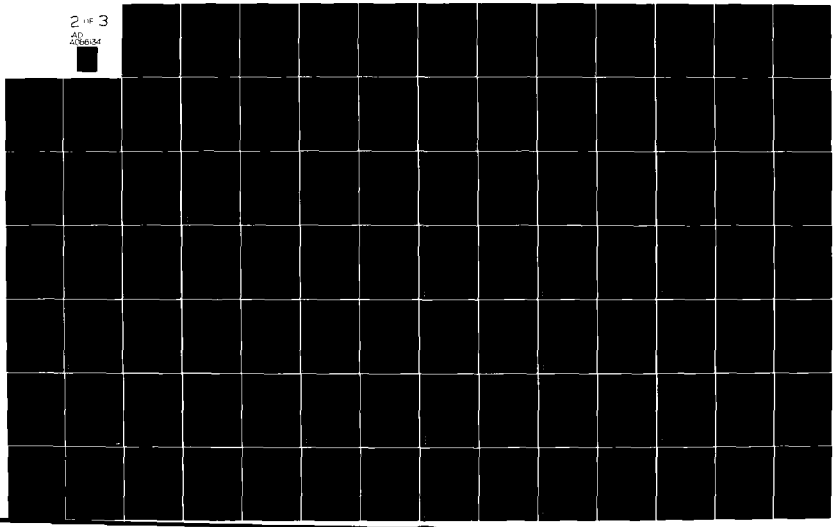


Fig. F.2 Algorithm for Initialization Step

AD-A086 134 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL

2 OF 3
AD A086134



There are two points worth to note as follows:

1. Because of parallel processing in the MAP-300, the synchronization between the AP and the AOM, and between the AP and the ADAM has to be considered. Memory is the common place which every processor has to access. Therefore, the synchronization is done by checking buffer availability as follows:
   A. If the first ADAM buffer is busy, CSPU goes to wait. Otherwise, goes to Step B.
   B. CSPU initiates AP. AP starts to read input samples from the first ADAM buffer, execute CVSD function and output results into the first AOM buffer.
   C. If the second ADAM buffer is busy, CSPU goes to wait, otherwise, goes to Step D.
   D. CSPU initiates AP. AP does the same function as Step B except it reads samples from the second ADAM buffer and outputs results to the second AOM buffer.
   E. Go to Step A.

   This structure is included in a function list.
2. In order to increase the flexibility of changing system parameters, there are some substitutive attributes inside the APS module. In a normal operation, CSPU gets values for those attributes from a host command and then binds them with the APS module before loading it into the APS memory. In order to decrease this overhead time, a "pre-bound" has to be done prior to real-time execution. The prebinding process will create a new APS module from old APS module by permanently binding the attributes. At the execution time, all normal operations of binding and buffer protection will be bypassed. Thus, the overhead time is decreased.

F.2.3 The Processing Step

In the processing step, each of three processors has a program. Among them, the data acquisition and data output programs, which are included in the Simple Notation for Array Processing library (SNAP library), are fairly simple. The actual CVSD algorithm is executed by the AP processor.

The real-time CVSD program can be divided into two parts as follows:

1. APU module of CVSD algorithm which is shown in Fig. F.3 will do the whole arithmetic operation described in the Section F.1. The input and output data are directed by APS module. The "hiding" technique is employed to reduced execution time.

2. The APS module of the CVSD algorithm which is shown in Fig. F.4 will produce address sequences of input and output data for APU module. Also, APS module acts as the controller of the real-time CVSD system. This AP control protocol is shown in Fig. F.5 and described as follows:
   * CPU starts APS-input by setting RI
   * APS sets the program counter for write routine and starts APS-output by setting RO.
   * APS starts APU by setting RA.
   * After producing addresses for input data, APS-input turns itself off by clearing RI.
   * When FI is cleared, APU leaves the loop.
   * After producing addresses for output data, APS-output turns itself off by clearing RO.
   * APU turns itself off by clearing RA.
   * CSPU is interrupted by both RAI (RA off) and DNI (AP done).

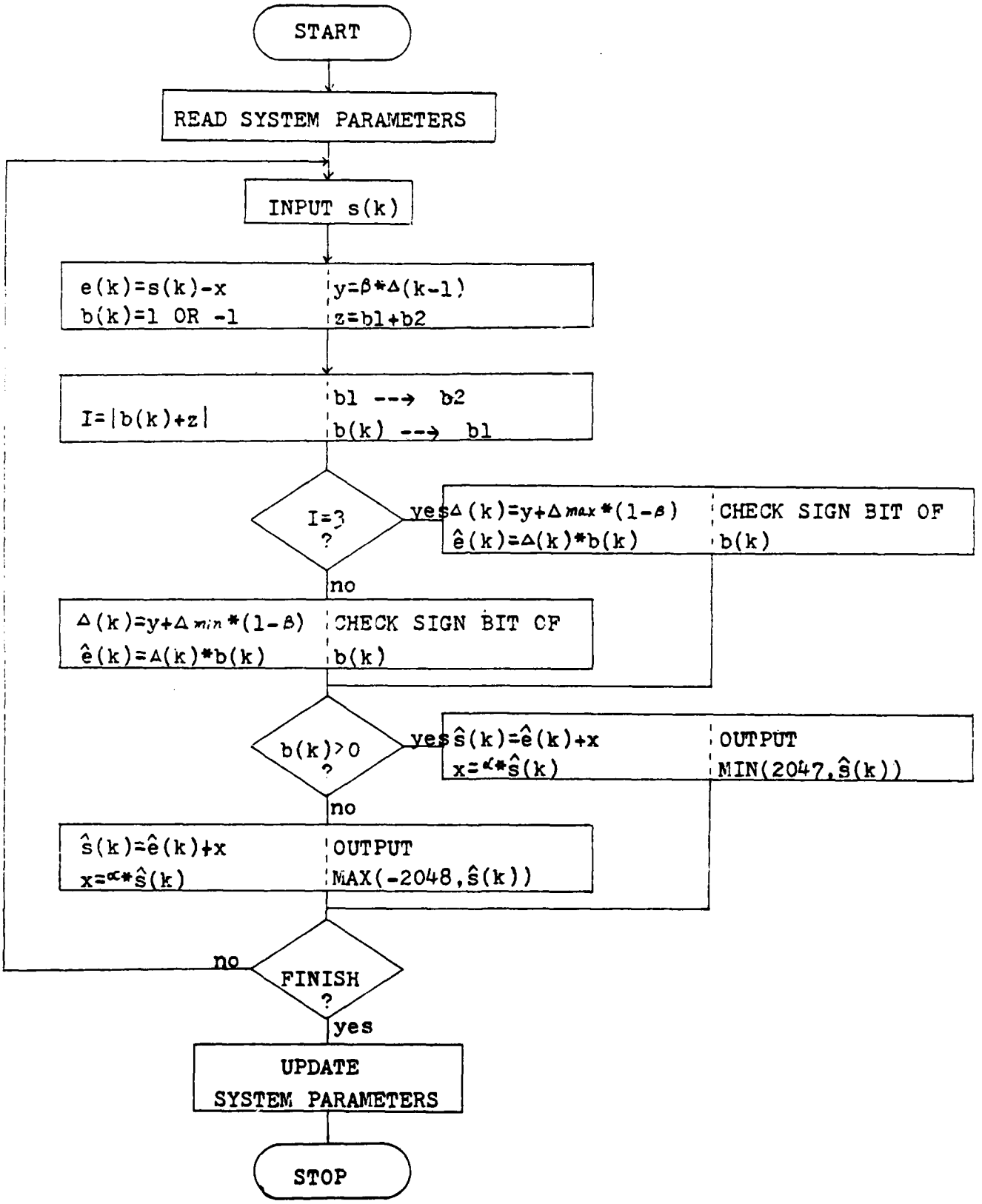
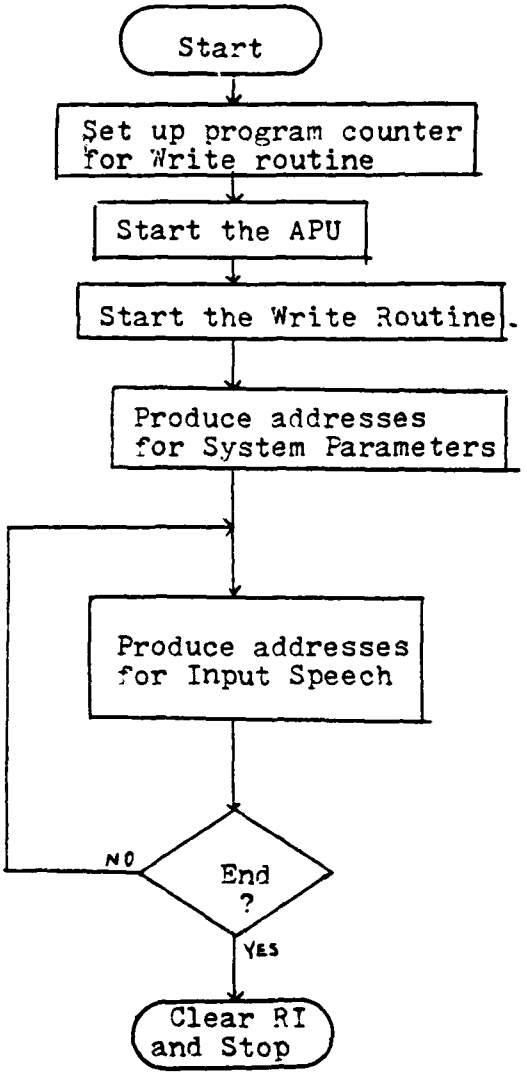
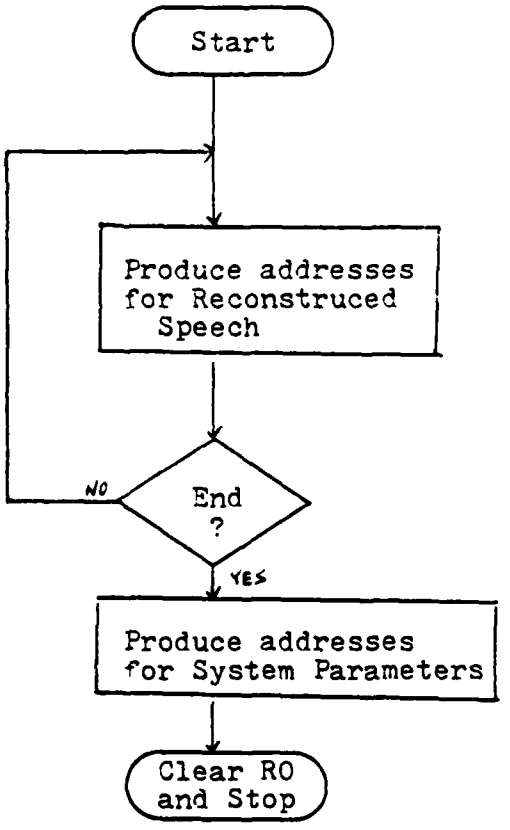


Fig. F.3 APU Flow Chart



(a)



(b)

Fig. F.4 APS Flow Chart
(a) APS-Input
(b) APS-Output

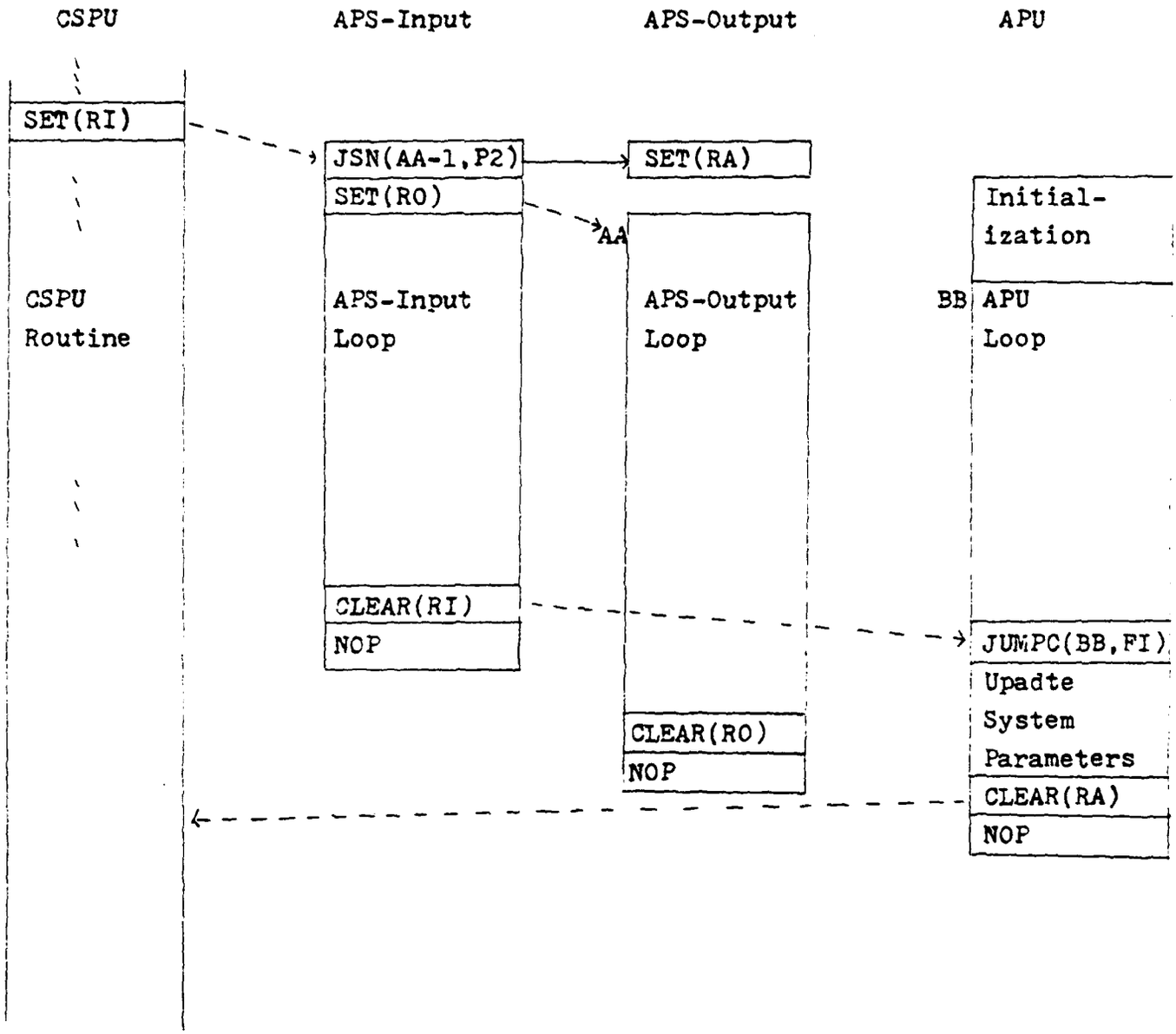


Fig. F.5 The AP Control Protocol

F.3 Conclusion

The execution time for the real-time CVSD algorithm ia approximately 12 micro-seconds per sample. So, this implementation can be used with a sampling rate up to 83 KH_z in the real-time mode. A set of parameters which gives a very good quality of speech is as follows:

$$\Delta_{max} = 750$$

$$\Delta_{min} = 25$$

$$\alpha = 0.94$$

$$\beta = 0.99$$

A two seconds sample speech is executed by this algorithm, and a 11.87 db signal-to-noise ratio performance is obtained.

F.4 Reference

[1] R. W. Schafer et. al., "Tandem Interconnection of LPC and CVSD Digital Speech Coders" Final Report to Defense Communications Agency, DCA Contract No. DCA 100-76-C-0073, November, 1977.

```
(ØØØØ1) *       MAP MODULE OF C V S D PROGRAM --- VERSION 1 --- FEB. 19, 198Ø
(ØØØØ2) *       ************************************************
(ØØØØ3) *       *                                              *
(ØØØØ4) *       *           REAL-TIME CVSD TRANSMITTER         *
(ØØØØ5) *       *                                              *
(ØØØØ6) *       ************************************************
(ØØØØ7) *
(ØØØØ8) *       DATA: 2/19/8Ø
(ØØØØ9) *       MAWLIN YEH
(ØØØ1Ø) *
(ØØØ11) *       PROGRAM NAME : CVSD.ND1
(ØØØ12) *
(ØØØ13) *       PROGRAM DESCRIPTION:
(ØØØ14) *
(ØØØ15) *       THIS MAP-FUNCTION SIMULATES AN ADAPTIVE DELTA-MODULATOR WHICH
(ØØØ16) *       USES A FIRST ORDER PREDICTOR WITH THE COEFFICIENT ALPHA AND
(ØØØ17) *       A FIRST ORDER ADAPTOR WITH THE COEFFICIENT BETA. THE VALUES OF
(ØØØ18) *       ALPHA AND BETA ARE TRANSFERED FROM HOST PROGRAM TO MAP SCALAR
(ØØØ19) *       TABLE.
(ØØØ2Ø) *
(ØØØ21) *       PROCEDURE OF FUNCTIONS:
(ØØØ22) *
(ØØØ23) *       1.READ IN SPEECH --- S(K)
(ØØØ24) *       2.CALCULATE THE PREDICTING ERROR --- E(K)=S(K)-P(K)
(ØØØ25) *       3.PASS E(K) THROUGH DELTA MODULATOR AND PRODUCE L(K)
(ØØØ26) *       4.GET THE LOGIC OUTPUT G(K)
(ØØØ27) *       5.CALCULATE ADAPTIVE STEP SIZE --- D(K)=BETA*D(K-1)+G(K)
(ØØØ28) *       6.CALCULATE RECONSTRUCTED PREDICTING ERROR --- EHAT(K)=L(K)*D(K)
(ØØØ29) *       7.CALCULATE RECONSTRUCTED SPEECH --- SHAT(K)=SHAT(K-1)+EHAT(K)
(ØØØ3Ø) *       8.UPDATE PREDICTOR'S PARAMETER = ALPHA*SHAT(K)
(ØØØ31) *
(ØØØ32) *       THIS PROGRAM PROVIDES UPPER LIMIT AND LOWER LIMIT OF OUTPUT
(ØØØ33) *       DATA. THESE TWO LIMITS ARE TRANSFERED FROM HOST PROGRAM.
(ØØØ34) *       THIS PROGRAM CONTAINS TWO SUB-PROGRAMS WHICH ARE CVSD FOR APU
(ØØØ35) *       AND V226ØD FOR APS. PASSING THROUGH MAP-SIMULATOR, CVSD TAKES
(ØØØ36) *       2.669 MICROSECONDS.  (LOOP TIME)
(ØØØ37) *
(ØØØ38) *       CALL TO THIS PROGRAM:
(ØØØ39) *
(ØØØ4Ø) *       CALL CVSD(Y,A,U)
(ØØØ41) *       OR
(ØØØ42) *       MAP=CVSD(Y,A,U)
(ØØØ43) *
(ØØØ44) *       WHERE    Y=OUTPUT BUFFER FROM MAP
```

```
                (0045) *                U=INPUT BUFFER TO MAP
                (0046) *                A=STARTING LOCATION OF THE FOLLOWING PARAMETERS
                (0047) *                1ST PARAMETER = UPPER LIMIT OF OUTPUT DATA
                (0048) *                2ND PARAMETER = LOWER LIMIT OF OUTPUT DATA
                (0049) *                3RD PARAMETER = LOWER LOGIC OUTPUT
                (0050) *                4TH PARAMETER = HIGHER LOGIC OUTPUT
                (0051) *                5TH PARAMETER = BETA
                (0052) *                6TH PARAMETER = THRESHOLD FOR LOGIC
                (0053) *                7TH PARAMETER = DELTA
                (0054) *                8TH PARAMETER = ALPHA
                (0055) *
                (0056) *
                (0057) *        CONTAINS FOR ASSEMBLY
                (0058) *
00000000        (0059)          MSS=0
00000794        (0060)          DMYS=S794                       DUMMY OUTPUT SPACE
00000001        (0061)          BUS1S=S0001                     BUS 1
00000288        (0062)          TOESPRT=S288                    TOP OF EXEC
000021FC        (0063)          CSPUSNOS=S21FC                  LOCATION OF NO-CSPU SUPPORT
00008E8         (0064)          AFDTS=S8E8                      STARTING ADDRESS OF AFDT
                (0065) *
                (0066) *        DISPATCH TABLE ENTRY
                (0067) *
000000F5        (0068)          FCB=245                         FCB=245
00008BA6        (0069)          #L=AFDTS+3*2*(FCB-128)          STARTING LOCATION OF AFDT FOR 245
                (0070) *
00BA6 001E4802  (0071)          ADDR        CVSDS(R7,1)        APU=CVSD
00BA8 001E4800  (0072)          ADDR        V2260DS(R7,1)       APS=V2260D
00BAA 001021FC  (0073)          ADDR        CSPUSNOS(,1,0)      NO CSPU
                (0074) *
                (0075) *        APU & APS MODULES
                (0076) *
00004800        (0077)          #L=S4800                        STARTING LOCATION OF CVSD AND V2260D
                (0078) *
                (0079) *        APU MODULE
                (0080) *
                (0081) *        FUNCTIONS:
                (0082) *
                (0083) *        ***** ADM1 *****                ***** ADM2 *****
                (0084) *        READ SCALARS                    READ SCALARS
                (0085) *        CALCULATE E(K)                  LOGIC OPERATION
                (0086) *        CALCULATE D(K)                  UPDATE LOGIC PARAMETERS
                (0087) *        CALCULATE EHAT(K) & SHAT(K)     NO OPERATION
                (0088) *        UPDATE PREDICTOR'S PARAMETERS   BOUND THE OUTPUT DATA
```

```
                    (0000) *
                    (0000)          EVEN                            START ON WORD BOUNDARY
     0400 0000      (0000)          DATA        CVSD$SA             STARTING ADDRESS
     0401 003B      (0000)          DATA        CVSD$SZ             SIZE
                    (0000) *
                    (0000) CVSD$    BEGIN       APU(CVSD)           START OF APU MODULE
              00000003 (0000)       @M=3                            MAP-300
              00000000 (0000)       @A=0                            CURRENT ADDRESS=0
                    (0000) *
                    (0000) *                                        ADM-1                   ADM-2
                    (0000) *                                        XXXXXXXXXXXXXXXXXX      XXXXXXXXXXXXXXXX
                    (0010) *        INITIALIZATION
                    (0010) *
  A00 0402 168000F6 (0012) CVSD$SA   K(1) \ MOV(IQA,A6)             ; R=1                   A6=UPPER LIMIT
  A01 0404 000000F7 (0013)          NOP \ MOV(IQA,A7)               ;                       A7=LOWER LIMIT
  A02 0406 00F30000 (0014)          MOV(IQA,A3) \ NOP               ;A3=LOWER LOGIC OUTPUT
  A03 0408 00921680 (0015)          MOV(R,A2) \ K(1)                ; A2=1
  A04 040A 00F60000 (0016)          MOV(IQA,A6) \ NOP               ; A6=HIGHER LOGIC OUTPUT
  A05 040C 000000EC (0017)          NOP \ MOV(IQA,M4)               ;--------               M4=BETA
  A06 040E 00F70000 (0018)          MOV(IQA,A7) \ NOP               ; A7=THRESHOLD          --------
  A07 0410 00EB00F0 (0019)          MOV(IQA,M3) \ MOV(R,M7)         ; M3=ALPHA              ------
  A08 0412 000000E8 (0110)          NOP \ MOV(IQA,M0)               ;                       M0=DELTA
  A09 0414 000000F3 (0111)          NOP \ MOV(IQA,A3)               ;                       A3=L(K-1)
  A0A 0416 000000F4 (0112)          NOP \ MOV(IQA,A4)               ;                       A4=L(K-2)
  A0B 0418 00F10000 (0113)          MOV(IQA,A1) \ NOP               ; A1=ALPHA*SHAT(K-1)
                    (0114) *
                    (0115) *        STARTING OF CVSD TRANSMITTER
                    (0116) *
  A0C 041A 00F000F0 (0117) LOOP     MOV(IQA,A0)                     ; SPEECH INPUT
  A0D 041C 49000400 (0118)          SUB(A0,A1) \ MUL(M0,M4)         ; E(K)=S(K)-SHAT(K-1)   BETA*DELTA
  A0E 041E 00154460 (0119)          MOV(ZERO,A5) \ ADD(A3,A4)       ; RESET A5              L(K-1)+L(K-2)
  A0F 0420 52A00000 (0120)          MOV(NULL),ADDST(A5,A2) \ NOP    ; DECIDE L(K)=+1 OR -1
                    (0121) *
  A10 0422 00000270 (0122)          NOP \ MOV(EXO),R(A3)            ;
  A11 0424 00540000 (0123)          MOV(EXI,A4) \ NOP               ;
  A12 0426 00900800 (0124)          MOV(R,EXO) \ MOV(P,EXO)         ; L(K)-->ADM2           BETA*DELTA-->ADM1
  A13 0428 45950094 (0125)          MOV(A5),ADD(A4,A5) \ MOV(R,A4)  ; I=L(K)+L(K-1)+L(K-2) L(K-1)-->L(K-2)
  A14 042A 00500053 (0126)          MOV(EXI,A0) \ MOV(EXI,A3)       ;                       L(K)--->L(K-1)
  A15 042C 77940058 (0127)          MOV(A4),MAXABS(A4,A7) \ MOV(EXI,EXO)  ; DECIDE LOGIC OUTPUT
  A16 042E 004D0000 (0128)          MOV(EXI,M5) \ NOP               ;
  A17 0430 00000000 (0129)          MOV(R,NULL) \ NOP               ;
                    (0130) *
  A18 0432 901E0038 (0131)          JUMPC(LOOP1,T1)                 ; IF T1=0 GO TO LOOP1
                    (0132) *
```

```
A19 #4834 43###26# (##133)          ADD(A#,A3) \ R(A3)              ; DELTA=BETA*DELTA+G(K)   L(K)=1 OR -1
A1A #4836 ######## (##134)          MOV(R,M#) \ NOP                ;
                  (##135) *
A1B #4838 842##### (##136) LOOP2    MUL(M#,M5) \ NOP               ; EHAT=DELTA;L(K)
A1C #483A ######## (##137)          MOV(R,EXO) \ NOP               ;
A1D #483C ######## (##138)          NOP \ MOV(EXI,M#)              ;                          RENEW M#=DELTA
A1E #483E 9#1F#### (##139)          JUMPC(LOOP4,T2)                ; IF L(K)=1 GO TO LOOP4
                  (##140) *
                  (##141) *         L(K)=-1 --- TEST IF OUTPUT DATA < LOWER LIMIT
                  (##142) *
A1F #4840 ######## (##143)          MOV(P,A5) \ NOP                ;
A2# #4842 41A##### (##144)          ADD(A5,A1) \ NOP               ; SHAT(K)=SHAT(K-1)+EHAT(K)
A21 #4844 ######## (##145)          MOV(R,EXO) \ NOP               ;
A22 #4846 ###E#851 (##146)          MOV(R,M6) \ MOV(EXI,A1)        ;
A23 #4848 85C#672# (##147)          MUL(M6,M3) \ MAX(A1,A7)        ; SHAT(K)=ALPHA+SHAT(K) SOUT=HIGHER VALU
A24 #484A ######9C (##148)          NOP \ MOV(R,OQ)                ;                          OUTPUT
A25 #484C ###1#### (##149)          MOV(P,A1) \ NOP                ; RENEW A1=SHAT(K)
A26 #484E 9#1D###C (##15#)          JUMPC(LOOP,FI)                 ; END OF ARRAY ?
A27 #485# 1###### (##151)           JUMP(LOOP3)                    ; END
                  (##152) *
                  (##153) *         L(K)=1 --- TEST IF OUTPUT DATA > UPPER LIMIT
                  (##154) ;
A28 #4852 ######## (##155) LOOP4    MOV(P,A5) \ NOP                ;
A29 #4854 41A##### (##156)          ADD(A5,A1) \ NOP               ; SHAT(K)=SHAT(K-1)+EHAT
A2A #4856 ######## (##157)          MOV(R,EXO) \ NOP               ;
A2B #4858 ###E#851 (##158)          MOV(R,M6) \ MOV(EXI,A1)        ;
A2C #485A 85C#6E2# (##159)          MUL(M6,M3) \ MIN(A1,A6)        ;SHAT(K)=#.94;SHAT(K), SOUT=LOWER VALUE
A2D #485C ######9C (##16#)          NOP \ MOV(R,OQ)                ;                          OUTPUT
A2E #485E ###1#### (##161)          MOV(P,A1) \ NOP                ; RENEW A1=SHAT(K)
A2F #486# 9#1D###C (##162)          JUMPC(LOOP,FI)                 ; END OF ARRAY ?
                  (##163) *
A3# #4862 #22#846# (##164) LOOP3    R(A1) \ MUL(M#,M7)             ;BACK UP PARAMETERS FOR NEXT BLOCK
A31 #4864 #89C#28# (##165)          MOV(R,OQ) \ R(A4)
A32 #4866 #####27C (##166)          NOP \ MOV(OQ),R(A3)
A33 #4868 ######9C (##167)          NOP \ MOV(R,OQ)
A34 #486A ######8C (##168)          NOP \ MOV(P,OQ)
A35 #486C 2#322#32 (##169)          CLEAR(RA)                      ; STOP APU
A36 #486E ######## (##17#)          NOP
A37 #487# 1###### (##171)           JUMP(#)                    ; CONTINUE
                  (##172) *
A38 #4872 46###26# (##173) LOOP1    ADD(A#,A6) \ R(A3)             ; DELTA=BETA+DELTA+G(K)  L(K)=1 OR -1
A39 #4874 ######## (##174)          MOV(R,M#) \ NOP                ;
A3A #4876 1#####1B (##175)          JUMP(LOOP2)                    ;
                  (##176) *
```

```
      00000038 (00177)          CVSD$SZ=.-CVSD$SA               ; COMPUTE THE SIZE OF THE MODULE
004870           (00178)          END
                 (00179)          EJECT
```

```
                    (00180) *         APS3-V2260D
                    (00181) *
                    (00182) *
                    (00183) *         THIS IS APS PROGRAM FOR CVSD.
                    (00184) *         IT EMPLOYES TWO INPUT-BUFFERS AND TWO OUTPUT-BUFFERS.
                    (00185) *         ALSO ONE CAN REPEAT THIS PROGRAM WITHOUT LOADING IT AGAIN.
                    (00186) *
                    (00187) *
                    (00188)           EVEN                              ; START ON WORD BOUNDARY
    0487B 0000048C2 (00189)           ADDR      V2260D$I                ; PTR TO CONSTR INSTR BLOCK
    0487A 00000488A (00190)           ADDR      V2260D$+2*V2260D$S       ; STARTING ADDR. OF SCALAR
    0487C 0001      (00191)           DATA      1                       ; ONE SCALAR
    0487D 004A      (00192)           DATA      V2260D$Z                ; SIZE
    0487E 0000048B2 (00193            ADDR      V2260D$A                ; CHAIN ANCHOR
                    (00194)           EVEN
                    (00195) *
                    (00196) *
                    (00197) V2260D$   BEGIN     APS(V2260D)             ; BEGIN OF THIS MODULE
                    (00198) *
                    (00199) *
A00 04880 00201540  (00200)           JSN(V2260D$5-1,P2)                ; SET APS-OUTPUT PC P2
A01 04882 02300030  (00201)           SET(R0)                           ; RUN APS-OUTPUT
                    (00202) *
                    (00203) *         SCALAR ADDRESSES
                    (00204) *
A02 04884 04C20000  (00205) V2260D$S  LOAD(BR0,MS$(1),L,TF)             ; UPPER LIMIT
A03 04886 068A0002  (00206)           ADD(BR0,2,TF)                     ; LOWER LIMIT
A04 04888 088A0002  (00207)           ADD(BR0,2,TF)                     ; LOWER LOGIC OUTPUT
A05 0488A 0A8A0002  (00208)           ADD(BR0,2,TF)                     ; HIGHER LOGIC OUTPUT
A06 0488C 0C8A0002  (00209)           ADD(BR0,2,TF)                     ; BETA
A07 0488E 0E8A0002  (00210)           ADD(BR0,2,TF)                     ; THRESHOLD
A08 04890 108A0002  (00211)           ADD(BR0,2,TF)                     ; ALPHA
A09 04892 128A0002  (00212)           ADD(BR0,2,TF)                     ; DELTA
A0A 04894 148A0002  (00213)           ADD(BR0,2,TF)                     ; L(K-1)
A0B 04896 168A0002  (00214)           ADD(BR0,2,TF)                     ; L(K-2)
A0C 04898 188A0002  (00215)           ADD(BR0,2,TF)                     ; ALPHA*SHAT(K-1)
                    (00216) *
                    (00217) *         1ST INPUT PROGRAM
                    (00218) *
A0D 0489A 1A710000  (00219) V2260D$1  LOAD(BR3,[1])                     ; SET STARTING ADDR.
A0E 0489C 1C500000  (00220)           LOAD(BR1,MS$)                     ; COUNTER--- SAMPLE SIZE - 1
A0F 0489E 1E320000  (00221)           SUB(BR3,MS$)                      ; BASE ADDRESS - SPACING
A10 048A0 20BA9006  (00222) V2260D$3  ADD(BR3,[9],TF)                   ; GIVE ADDR.
A11 048A2 22191081  (00223)           SUBL(BR1,1),JUMPP(V2260D$3)       ; CHECK END ?
```

```
  A12 0048A4 242000031 (00224)          CLEAR(RI)                    ;STOP APS-INPUT
  A13 0048A6 26000002 (00225)           NOP(0)
  A14 0048A8 28000006 (00226)           JUMP(0)
                     (00227) *
                     (00228) *          OUTPUT PROGRAM
                     (00229) *
  A15 0048AA 2A300032 (00230)           SET(RA)                      ; RUN APU
  A16 0048AC 2C400C (00231) V2260DS5    LOAD(BW0,[0])                ; STARTING ADDRESS
  A17 0048AE 2E500000 (00232)           LOAD(BW1,MS$)                ; COUNTER--- SAMPLE SIZE -1
  A18 0048B0 38020006 (00233)           SUB(BW0,MS$)                 ; BASE ADDRESS - SPACING
  A19 0048B2 328A0006 (00234) V2260DS6  ADD(BW0,[0],TF)              ; OUTPUT ADDRESS
  A1A 0048B4 34111981 (00235)           SUBL(BW1,1),JUMPP(V2260DS6)  ; END ?
  A1B 0048B6 36010011 (00236)           MOVB(BW0,BR0,TF)             ; BACKUP PARAMETERS
  A1C 0048B8 38020002 (00237)           SUB(BW0,2,TF)
  A1D 0048BA 3A020002 (00238)           SUB(BW0,2,TF)
  A1E 0048BC 3C020002 (00239)           SUB(BW0,2,TF)
  A1F 0048BE 3E200030 (00240)           CLEAR(RO)
  A20 0048C0 40000020 (00241)           NOP(0)
                     (00242) *
                     (00243) *
            0000480A (00244) V2260DSA=0C                           ; CHAIN ANCHOR
                     (00245) *
      0048C2         (00246)           END
                     (00247)           EVEN
                     (00248) *
                     (00249) *          STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                     (00250) *
      0048C2 00000000 (00251) V2260DSI  DATA      4F'0.0'
       ...
                     (00252) *
            00000048 (00253) V2260DSZ=0L-V2260DS                   ; COMPUTE MODULE SIZE
                     (00254) *
                     (00255) *        TOP OF EXEC
                     (00256) *
            0000048CA (00257)           TOES=0L
            00000288 (00258)           0L=TOESPRT
      00288 00010048CA (00259)          ADDR      TOES(,BUS1S)
      0028A          (00260)           END
```

```
AFDT$:      ##8E8 (###64) (###69)
BUS1$:      #3##1 (###61) (##259)
CSPU$NOS:   #21FC (###63) (###73)
CVSD$:      #48#2 (###71) (###94)
CVSD$SA:    ##### (###91) (##1#2) (##177)
CVSD$SZ:    ###3B (###92) (##177)
DMY$:       ##794 (####6#)
FCB:        ###F5 (###68) (###69)
LOOP:       ####C (##117) (##15#) (##162)
LOOP1:      ###3B (##131) (##173)
LOOP2:      ###1B (##136) (##175)
LOOP3:      ###3# (##151) (##164)
LOOP4:      ###28 (##139) (##155)
MS$:        ##### (###59) (##2#5) (##22#) (##221) (##232) (##233)
TOE$:       #48CA (##257) (##259)
TOE$PRT:    ##288 (###62) (##258)
V226#D$:    #486# (###72) (##19#) (##197) (##253)
V226#D$1:   ####D (##219)
V226#D$3:   ###1# (##222) (##223)
V226#D$5:   ###16 (##2##) (##231)
V226#D$6:   ###19 (##234) (##235)
V226#D$A:   #48B2 (##193) (##244)
V226#D$I:   #48C2 (##189) (##251)
V226#D$S:   ####2 (##19#) (##2#5)
V226#D$Z:   ###4A (##192) (##253)

LINES WITH ERRORS:    #  (MAP VERSION 8##1#1.1#)   E-    #
```

```
CVSDHS.FTN      /TR:BLOCKS/VR

        C       ************************************************
        C       *                                              *
        C       *       CVSD HOST SUPPORT PROGRAM              *
        C       *                                              *
        C       ************************************************
        C
        C       DATE: 11/14/79
        C       MAWLIN YEH
        C
        C       CVSD HOST SUPPORT PROGRAM
        C
        C       CALL CVSD(Y,A,U)
        C       OR
        C       MAP=CVSD(Y,A,U)
        C
        C       WHERE:  Y=OUTPUT BUFFER
        C               A=STARTING LOCATION OF SCALARS
        C               U=INPUT BUFFER
        C
####1           INTEGER FUNCTION CVSD(Y,A,U)
####2           INTEGER Y,A,U,FCBGN
####3           CVSD=FCBGN(245,Y,A,U,#,#,#,#,#,5)
####4           RETURN
####5           END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	SCODE1	000062	25	RW,I,CON,LCL
2	SPDATA	000014	6	RW,D,CON,LCL
3	SIDATA	000038	12	RW,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
CVSD	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
A	I*2	F-000004*	U	I*2	F-000006*	Y	I*2	F-000002*						

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 000126 43

NO FPP INSTRUCTIONS GENERATED

CVSDHS,CVSDHS/-SP=CVSDHS

```
+CVSDBN.FTN      /TR:BLOCKS/WR

C       ************************************************
C       *                                              *
C       *                  C V S D                     *
C       *              IN REAL TIME MODE               *
C       *                                              *
C       ************************************************
C
C       PROGRAM: REALCVSD.FTN
C
C       DATE: 2-19-8#
C       MAWLIN YEH
C
C       THIS PROGRAM DOES NOT HAVE THE SOURCE CODING ROUTINES AND CVSD-RECEIVER.
C
C       ITS FUNCTION IS DESCRIBED AS FOLLOWS:
C
C       A CVSD TRANSMITTER IS SIMPLY A DELTA MODULATOR WITH AN ADAPTIVE
C       STEP SIZE. IN ORDER TO MINIMIZE THE DIFFERENCE BETWEEN ADJACENT
C       SAMPLES, THE INPUT SIGNAL S(K) IS NORMALLY OVERSAMPLED. TYPICALLY,
C       THE SAMPLING RATE IS 16K BITS PER SECOND.
C       THE TRANSMITTER QUANTIZES THE DIFFERENCE E(K) BETWEEN AN INPUT
C       SPEECH SAMPLE AND ITS ESTIMATE WHICH IS PREDICTED BY A FIRST
C       ORDER PREDICTOR.
C               E(K) = S(K) - ALPHA * SHAT(K-1)
C       WHERE SHAT(K-1) IS THE RECONSTRUCTED SPEECH OF THE INPUT SPEECH
C       SAMPLE AT TIME K-1.
C       THEN THE TRANSMITTER OUTPUTS A SINGLE BIT, B(K), FOR EACH CORRESPONDING
C       INPUT SAMPLE WHERE
C                       |->     1       IF E(K) >= #
C               B(K)=   |
C                       |->     -1      IF E(K) <  #
C       THE ADAPTIVE STEPSIZE LOGIC DRIVES ITS OUTPUT DELTA(K) FROM THE
C       BIT STREAM B(K) AND AN APPROPRIATE INITIAL STATE,
C               DELTA(K) = BETA * DELTA(K-1) + G(K)
C       WHERE
C                       |->     DELTA-MAX * (1-BETA)    IF B(K)=B(K-1)=B(K-2)
C               G(K)=   |
C                       |->     DELTA-MIN * (1-BETA)    OTHERWISE
C       THE ESTIMATE OF THE DIFFERENCE SIGNAL IS GIVEN BY
C               EHAT(K) = B(K) * DELTA(K)
C       THEN THE RECONSTRUCTED SPEECH IS COMPUTED FROM
C               SHAT(K) = ALPHA * SHAT(K-1) + EHAT(K)
C
C       THE SET OF PARAMETERS WHICH CAN GET THE BEST RESULT FOR CHANNEL
C       NOISE ARE AS FOLLOWS:
C               ALPHA = #.94
C               BETA = #.99
C               DELTA-MAX = 75#
C               DELTA-MIN = 25
C
C       THE WAY TO SET UP CVSD IN MAP-3## IS AS FOLLOWS:
C       1. TO PRODUCE A NEW SNAP-LIBRARY:
C               F4P CVSDHS=CVSDHS
C               LBR SNPLIB/IN=CVSDHS
C       2. TO PRODUCE A CVSD TASK FILE:
C               F4P REALCVSD=REALCVSD
```

```
C                   TSK
C                   REALCVSD/CP/FP=REALCVSD,SNPLIB/LB
C                   /
C                   UNITS=10
C                   ASG=IN:4
C                   ASG=ML:6
C                   ASG=TI:5:7
C                   //
C           3. TO COMPILE CVSD-ASSEMBLY PROGRAM:
C                   RUN MAPASM
C                   CVSD.ND1
C                   CVSD.R35
C                   CVSD.LST
C                   $1,8000
C           4. TO LOAD MAP-300
C                   RUN MPLD
C                   QL
C                   RUN MPLD
C                   CVSD.R35
C           5. TO EXECUTE CVSD:
C                   ASN DM:=IN:
C                   ASN DM:=ML:
C                   RUN REALCVSD
C
0001              REAL SCAL(15)
0002              INTEGER BSIN(2),BSHTR(2),FL1
0003              INTEGER IOS2,IDAOM,AOM,IDADM,ADM,CHAN1(16)
0004              INTEGER REAL,CMPLX,FIXED,LONG,SHORT,CONTIG,ALT
0005              DATA BSIN,BSHTR/1,2,3,4/
0006              DATA FL1/1/
0007              DATA IOS2,AOM,ADM,IDAOM,IDADM/2,22,23,13,14/
0008              DATA CHAN1/1,-1,14*0/
0009              DATA LONG,SHORT,REAL,CMPLX,FIXED,CONTIG,ALT/0,1,0,1,2,1,2/
0010              TYPE *,' *****     C V S D     *****'
0011              TYPE *,'        U. OF NOTRE DAME'
C---              INFORMATION
0012    999       TYPE *,' TYPE IN INPUT SPEECH BUFFER SIZE?'
0013              READ(4,*)SIZE
0014              ISIZE=SIZE
0015              WRITE(6,*)' SIZE=',ISIZE
0016              TYPE *,' ENTER VALUES: UPPER LIMIT OF SHAT, LOWER LIMIT OF SHAT,'
0017              TYPE *,' DELTA MAX., DELTA MIN.,BETA,THRESHOLD,ALPHA'
0018              READ(4,*)SCAL(1),SCAL(2),DELTM,SCAL(8),SCAL(5),THRE,
                2 SCAL(7)
0019              SCAL(3)=(1.-SCAL(5))*SCAL(8)
0020              SCAL(4)=(1.-SCAL(5))*DELTM
0021              SCAL(6)=THRE-0.5
0022              WRITE(6,*)' UPPER LIMIT =',SCAL(1),'LOWER LIMIT =',SCAL(2)
0023              WRITE(6,*)' UPPER LOGIC =',SCAL(4),'LOWER LOGIC =',SCAL(3)
0024              WRITE(6,*)' DELTA-MAX =',DELTM,'DELTA-MIN =',SCAL(8)
0025              WRITE(6,*)' ALPHA =',SCAL(7),'BETA =',SCAL(5)
0026              WRITE(6,*)' THESHOLD =',THRE
0027              SCAL(1)=SCAL(1)/16.**3
0028              SCAL(2)=SCAL(2)/16.**3
0029              SCAL(3)=SCAL(3)/16.**3
0030              SCAL(4)=SCAL(4)/16.**3
```

```
#031              SCAL(8)=SCAL(8)/16.**3
#032              TYPE *,'** TYPE IN SAMPLING FREQ.'
#033              READ(4,*)FREQ
#034              WRITE(6,*)' SAMPLING FREQUENCY =',FREQ
#035              CLOSE(UNIT=4)
        C---      DEFINE BUS IDENTIFIERS
#036              IBUS1=64
#037              IBUS2=128
#038              IBUS3=192
        C
        C         MAP COFINGURATION BEGINS HERE
#039              M=MPOPN(1)
#04#              DO 1## I=1,2
#041              SSIZE=(I-1)*SIZE
#042              M=MPCLB(IBUS3+BSIN(I),#.#+SSIZE,SIZE,REAL,CONTIG,SHORT)
#043              M=MPCLB(IBUS2+BSHTR(I),#.#+SSIZE,SIZE,REAL,CONTIG,SHORT)
        C---      INITIALIZE THESE BUFFERS
#044              M=VSMA1(BSIN(I),#,BSIN(I),#)
#045              M=VSMA1(BSHTR(I),#,BSHTR(I),#)
        C
#046    1##       CONTINUE
#047              M=MPCLB(IBUS1+IDAOM,3####.#,512.#,FIXED,CONTIG,LONG)
#048              M=MPCLB(IBUS1+IDADM,31###.#,512.#,FIXED,CONTIG,LONG)
#049              M=MPCLB(IBUS1+11,10246.,100.,2,1,#)
        C         CREATE PREBOUND FUNCTION
#05#              M=MPCBF(11,25#)
#051              M=CVSD(BSHTR(1),5#,BSIN(1))
#052              M=CVSD(BSHTR(2),5#,BSIN(2))
#053              M=MPTBF(#)
        C         LOAD THE ADAM AND AOM
#054              M=AOMID(IDAOM,FREQ,#,BSHTR(2),BSHTR(1),2)
#055              M=ADMSD(IDADM,FREQ,1.#,#,CHAN1,BSIN(2),BSIN(1))
#056              M=MPLDS(AOM,IOS2,IDAOM)
#057              M=MPLDS(ADM,IOS2,IDADM)
#058              M=MPWST(5#,SCAL(1),15,1)
        C
        C---      FUNCTION LIST
#059              M=MPBFL(FL1)
#06#              M=MPWT(2,BSIN(1))
#061              M=MPXBF(25#)
#062              M=MPWT(2,BSIN(2))
#063              M=MPXBF(251)
#064              M=MPEFL(FL1)
#065              M=MPRAA(ADM,#.#,AOM,#.#)
#066              M=MPWHL(#,#,FL1)
        C---      GO INTO WAIT STATE
#067              TYPE *,'** REAL TIME CVSD IS RUNNING **'
#068    5##       TYPE *,' TYPE IN #-STOP OR 1-CHANGE PARAMETERS'
#069              READ(5,*)LDAT
#07#              IF(LDAT .NE. # .AND. LDAT .NE. 1)GO TO 5##
#071              M=MPSAA(AOM)
#072              M=MPSAA(ADM)
#073              M=MPCLS(#)
#074              IF(LDAT .EQ. 1)GO TO 999
#075              STOP
#076              END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	SCODE1	002522	681	RW,I,CON,LCL
2	SPDATA	000762	249	RW,D,CON,LCL
3	SIDATA	000320	104	RW,D,CON,LCL
4	SVARS	000240	80	RW,D,CON,LCL
5	STEMPS	000002	1	RW,D,CON,LCL

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ADM	I*2	4-000116	ALT	I*2	4-000174	AOM	I*2	4-000112	CMPLX	I*2	4-000162	CONTIG	I*2	4-000172
DELTM	R*4	4-000204	FIXED	I*2	4-000164	FL1	I*2	4-000104	FREQ	R*4	4-000214	I	I*2	4-000230
IBUS1	I*2	4-000220	IBUS2	I*2	4-000222	IBUS3	I*2	4-000224	IDADM	I*2	4-000114	IDAOM	I*2	4-000110
IOS2	I*2	4-000106	ISIZE	I*2	4-000202	LDAT	I*2	4-000236	LONG	I*2	4-000166	M	I*2	4-000226
REAL	I*2	4-000160	SHORT	I*2	4-000170	SIZE	R*4	4-000176	SSIZE	R*4	4-000232	THRE	R*4	4-000210

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
BSHTR	I*2	4-000100	000004	2	(2)
BSIN	I*2	4-000074	000004	2	(2)
CHANI	I*2	4-000120	000040	16	(16)
SCAL	R*4	4-000000	000074	30	(15)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	500	1-002340	999	1-000076				

FUNCTIONS AND SUBROUTINES REFERENCED

ADMSD AOMID CLOSS CVSD MPBFL MPCBF MPCLB MPCLS MPEFL MPLDS MPOPN MPRAA MPSAA MPTBF MPWHL MPWST
MPWT MPXBF VSMAI

TOTAL SPACE ALLOCATED = 004266 1115

CVSDBN,CVSDBN/-SP=CVSDBN

APPENDIX G
PROGRAM LISTINGS FOR REAL TIME IMPLEMENTATION OF PARC ON MAP-300

This appendix contains the program listings of all the modules in the real time implementation of the PARC algorithm on the MAP-300 array processor. There are twenty modules in all, some in the PDP-11, and others in the various processors of the MAP 300. Ten modules execute in the MAP-300. Nine of these have host support programs in the PDP-11. And finally, there is the main program in the PDP-11 which sets up and initializes the real time PARC operation on the MAP-300.

The following is a list of these programs and the processors they execute on.

1.	PARC Mainline	PDP-11
2-10.	Host Support Subroutines	PDP-11
11.	Receiver Update Module	CSPU/MAP-300
12.	Transmitter Update Module	CSPU/MAP-300
13.	Initialization Module	CSPU/MAP-300
14.	Pitch Computation Module	AP/MAP-300
15.	PARC Transmitter Module	AP/MAP-300
16.	PARC Receiver Module	AP/MAP-300
17.	Encoder Module	CSPU/MAP-300
18.	IOS Simulator	CSPU/MAP-300
19.	Decoder Module	CSPU/MAP-300
20.	Digital I/O Module	IOS/MAP-300

Operator Sequence to Run PARC at DCA

The two MAP's at DCA have different logical device names, MP and MA. There is a different task images corresponding to each MAP, called PARP.TSK and PARA.TSK. The task names for these programs are NDP and NDA.

To run the real time PARC algorithm, the following steps must be followed for each MAP. For the steps shown below, the devices are called MX. The underlined text is the computer response at the terminal. The rest of the text is the operator input.

```
> LOA  MX:          (Load the map device)

> INS  MXLD/TASK = ... XLO      (Install map loader program)

> XLO               (Load the map executive including PARC routines.
                     The executive is called NDQL)

> INS PARX          (Install the initialization routines for PARC)

> NDX               (Run the initialization program)

MODE ?(1 - Internal Loopback, 2-Full Duplex)

2                   (Mode Selection)
```

These steps for setting up and executing the PARC algorithm are combined into an indirect command file called PARC.CMD. If the indirect command file is used, the operator only needs to run the program (NDX) and select the mode.

```
PARC96.FTN      /TR:BLOCKS/WR

        C       ********************************************************
        C       *                                                      *
        C       *       PITCH EXTRACTION ADAPTIVE RESIDUAL CODER       *
        C       *                       P A R C                        *
        C       *                                                      *
        C       ********************************************************
        C
        C                                                  MAR 2#,198#
        C
        C       REAL TIME IMPLEMENTATION INCLUDING ENCODER, DECODER,
        C       IOS LOOP-BACK.
        C
#0#1            REAL SCAL3(26),SCAL1(15),SCAL2(29),A1(8#),A2(8#)
#0#2            REAL OUTSCA(2#),EXPN(2#),T(2#),B(2#),HQBETA
#0#3            INTEGER ADM1,ADM2,SAMP,TXVHAT,T1SHAT,T2SHAT,TXPARA
#0#4            INTEGER INIS(14),KBLK,AOM1,AOM2,AOM3,AOM4
#0#5            INTEGER RCVHAT,R1SHAT,R2SHAT,RCPARA,BOUT1,BOUT2,TBLK,FBLK
#0#6            INTEGER RCVHT2
#0#7            INTEGER IOSPGM(2##),SIZE
#0#8            INTEGER ENCT(3#),ENCB(97),DECT(256),DECB(128)
#0#9            INTEGER PWBTA(8),DECTAB(2,14),ISEL(3),OFFSET
#0#1#           INTEGER ENTRB,ENTRT,DCRCT,DCRCB
#0#11           INTEGER QTR1,QTR2,BTR1,BTR2,QRC1,QRC2,BRC
#0#12           INTEGER ITBTR1,ITBTR2,ITBRC1,ITBRC2
#0#13           INTEGER ADM,AOM,IOS,IOS2
#0#14           INTEGER IDADM,IDAOM,IDIOS
#0#15           INTEGER IFSEL,CHAN1(16)
#0#16           INTEGER REAL,CMPLX,FIXED,LONG,SHORT,CONTIG,ALT
#0#17           DATA IBUS1,IBUS2,IBUS3/64,128,192/
#0#18           DATA ADM1,ADM2,BOUT1,BOUT2/1,2,3,4/
#0#19           DATA SAMP,TXVHAT,T1SHAT,T2SHAT,TXPARA/5,6,7,8,9/
#0#2#           DATA RCVHAT,R1SHAT,RCVHT2,RCPARA/1#,11,12,13/
#0#21           DATA ENTRB,ENTRT,DCRCT,DCRCB/14,15,16,17/
#0#22           DATA QTR1,QTR2,QRC1,QRC2,BTR1,BTR2,BRC/18,19,2#,21,22,23,24/
#0#23           DATA IDADM,IDAOM,IDIOS/27,28,29/
#0#24           DATA CHAN1/1,-1,14*#/
#0#25           DATA LONG,SHORT,REAL,CMPLX,FIXED,CONTIG,ALT/#,1,#,1,2,1,2/
#0#26           DATA AOM1,AOM2,AOM3,AOM4/31,32,33,34/
#0#27           DATA ADM,AOM,IOS,IOS2/23,22,16,2/
#0#28           DATA ITBTR1,ITBTR2,ITBRC1,ITBRC2,IFSEL/88,91,94,96,8#/
#0#29           DATA IFPQL,IFPRC,IFPIOS,IFMODE,IFSYNC,IFRCVR/1,2,3,4,5,6/
#0#3#           DATA IFSYN1,IFSYN2,IFREC1,IFREC2/7,8,9,1#/
        C
#0#31           DATA ENCT/3,7,#,#,2,5,3,11,5,47,
                1          6,95,7,191,2,1,3,3,5,15,
                2          6,31,7,63,7,127,7,255,7,255/
        C
#0#32           DATA DECTAB/14,-2#,#,1,5,2,13,3,61,4,
                1            125,5,253,6,4,7,12,8,6#,9,
                2            124,1#,252,11,254,-12,255,-12/
        C
#0#33           DATA ENCB/7,11,13,14,15,19,21,22,23,25,26,27,28,29,3#,31,
                1         35,37,38,39,41,42,43,44,45,46,47,49,5#,51,52,53,
                2         54,55,56,57,58,59,6#,61,62,63,67,69,7#,71,73,74,
                3         75,76,77,78,79,81,82,83,84,85,86,87,88,89,9#,91,
                4         92,93,94,95,97,98,99,1##,1#1,1#2,1#3,1#4,1#5,1#6,
```

```
                5                107,108,
                6        109,110,111,112,113,114,115,116,117,118,119,120,
                7                121,122,123,124,125/
        C
0034            DATA PHBTA/0,1,2,4,8,16,32,64/
        C
        C       INFORMATION AND PARAMETERS
        C
0035    999     OPEN(UNIT=4,NAME='PARA.DAT',SHARED,TYPE='OLD',READONLY)
        D       TYPE *,' TYPE IN FILTER PARAMETERS : A, B?'
0036            READ(4,*)SCAL1(2),SCAL1(3)
        D       TYPE *,' TRANSFER BLOCK SIZE (MULTIPLES OF 2) ?'
0037            READ(4,*)TBLK
        D       TYPE *,' FILTERING BLOCK SIZE?'
0038            READ(4,*)FBLK
        D       TYPE *,' CORRELATION BLOCK SIZE (MULTIPLES OF 4) ?'
0039            READ(4,*)KBLK
0040            INIS(3)=FBLK-1
0041            INIS(5)=KBLK-1
0042            INIS(7)=TBLK
0043            INIS(14)=TBLK
0044            FFBLK=FBLK
0045            RKBLK=KBLK
0046            TTBLK=TBLK
0047            SCAL1(1)=TTBLK/2.-0.5
0048            SCAL1(4)=FFBLK-0.5
0049            SCAL1(5)=RKBLK/4.-0.5
        C
0050            SCAL3(5)=TTBLK-0.5
        D       TYPE *,' RUN LENGTH, BIT # FOR LEVEL 1 AND BIT # FOR RUN LENTH CODE ?'
0051            READ(4,*)SCAL2(5),BIT1,BITLNT
0052            SCAL2(16)=SCAL2(5)*BIT1-BITLNT
0053            NRUN=SCAL2(5)-1
0054            SCAL2(15)=SCAL2(5)
        C
        D       TYPE *,' PITCH PERIOD SEARCH RANGE: FROM ? TO ?'
0055            READ(4,*)ITSTRT,ITEND
0056            TSTRT=ITSTRT
0057            TEND=ITEND
0058            INIS(4)=ITSTRT-1
0059            INIS(10)=ITSTRT
0060            SCAL1(6)=TEND-TSTRT+0.5
0061            SCAL1(8)=TEND-0.5
0062            SCAL1(10)=TSTRT
        C
        D       TYPE *,' BIT BLOCK SIZE, # BITS NULL CODE, GAP SIZE, # BITS PHONEY BETA ?'
0063            READ(4,*)SCAL2(2),SCAL2(21),IGAP,SCAL2(6)
0064            INIS(6)=IGAP
0065            INIS(9)=IGAP
0066            INIS(13)=IGAP
0067            GAP=IGAP
0068            SCAL2(7)=GAP-0.5
0069            SCAL3(19)=SCAL2(7)
        C
        D       TYPE *,' GAP THRESHOLD & FILTER THRESHOLD ?'
0070            READ(4,*)INIS(1),INIS(2)
```

```
C
D       TYPE *,' VALUES FOR RMSMIN,ALAD,ALP,SMIN,G,AINV,INIT STATE?'
0071            READ(4,*)RMSMIN,ALAD,ALP,SMIN,G,AINV,STATE
0072            N=4
0073            SCAL2(18)=AINV*(1/ALAD-1)
0074            SCAL2(19)=ALAD
0075            SCAL2(20)=N-2.5
0076            SCAL2(3)=RMSMIN/16.**3
0077            SCAL2(4)=STATE
0078            SCAL2(8)=RMSMIN/16.**3
0079            SCAL2(9)=N-2.5
0080            SCAL2(10)=1-ALP
0081            SCAL2(11)=ALP
0082            SCAL2(12)=SMIN
0083            SCAL2(13)=G
0084            SCAL3(6)=RMSMIN/16.**3
0085            SCAL3(7)=STATE
0086            SCAL3(9)=RMSMIN/16.**3
0087            SCAL3(11)=ALAD
0088            SCAL3(10)=N-2.5
0089            SCAL3(15)=1-ALP
0090            SCAL3(16)=ALP
0091            SCAL3(12)=SMIN
0092            SCAL3(14)=G
0093            SCAL3(17)=AINV*(1/ALAD-1)
C
D       TYPE *,' # OF QUANTIZER LEVELS?'
0094            READ(4,*)IK
0095            IL=IK/2
0096            SCAL2(14)=IL
0097            ILL=IL+1
D       TYPE *,' OUTPUT SCALING FACTORS FROM 1 TO',ILL
0098            READ(4,*)(OUTSCA(J),J=1,ILL)
D       TYPE *,' EXPANSION/CONTRACTION FACTORS FROM 1 TO',ILL
0099            READ(4,*)(EXPN(J),J=1,ILL)
D       TYPE *,' # BITS  CORRESPONDING TO EACH LEVEL FROM 1 TO',IK
0100            READ(4,*)(B(I),I=1,IK)
0101            K1=9
0102            K11=9
0103            A1(K1)=(OUTSCA(1)+OUTSCA(2))/2
0104            T(1)=A1(K1)
0105            A1(K1+1)=OUTSCA(1)
0106            A1(K1+2)=EXPN(1)
0107            A1(K1+3)=B(1)
0108            A1(K1+4)=B(1)
0109            A2(K11)=EXPN(1)
0110            A2(K11+1)=OUTSCA(1)
0111            DO 40 I=2,IL
0112            T(I)=(OUTSCA(I)+OUTSCA(I+1))/2
0113            K11=K11+2
0114            A2(K11)=EXPN(I)
0115            A2(K11+1)=OUTSCA(I)
0116            K1=K1+5
0117            A1(K1)=T(I)
0118            A1(K1+1)=OUTSCA(I)
0119            A1(    2)=(    (I)
```

```
0128            A1(K1+3)=B(I)
0121     48     A1(K1+4)=B(I+IL)
0122            K1=K1+5
0123            A1(K1)=OUTSCA(ILL)
0124            A1(K1+1)=EXPN(ILL)
0125            A1(K1+2)=B(ILL)
0126            A1(K1+3)=B(ILL+IL)
0127            K11=K11+2
0128            A2(K11)=EXPN(ILL)
0129            A2(K11+1)=OUTSCA(ILL)
0130            DO 44 I=1,IL
0131            K11=K11+2
0132            A2(K11)=A2(9+2*I)
0133            A2(K11+1)=-A2(9+2*I+1)
0134     44     CONTINUE
        C
        D       TYPE *,' UPPER LIMIT OF BETA?'
0135            READ(4,*)UBETA
0136            SCAL1(11)=-UBETA
0137            SCAL1(12)=UBETA
0138            SCAL3(13)=SCAL1(12)
        C
        D       TYPE *,' # OF QUANTIZATION LEVELS FOR BETA?'
0139            READ(4,*)QBETA
0140            IQBETA=QBETA
0141            HQBETA=IQBETA/2
0142            SCAL1(13)=HQBETA/(UBETA*(2.**15))
0143            SCAL1(14)=HQBETA/(UBETA*(2.**24))
0144            SCAL1(15)=1./SCAL1(14)
0145            SCAL3(8)=1./SCAL1(13)
        C
        D       TYPE *,' SAMPLING FREQUENCE FOR ADAM ?'
0146            READ(4,*)FREADM
        C
        D       TYPE *,' INVERSE GAIN FACTOR FOR MASKING LSBS ?'
0147            READ(4,*) SCAL3(24)
0148            SCAL3(24)=1./(SCAL3(24)*2.**13)
        D       TYPE *,' MAX. Q-BUFFER LENGTH ( < 500 ) ?'
0149            READ(4,*) SCAL3(25)
        D       TYPE *,' GAIN FACTOR AT ADAM INPUT ?'
0150            READ(4,*) SCAL3(26)
0151            SCAL3(26)=SCAL3(26)/SCAL3(24)
        C
0152            SCAL1(7)=1./(2.**15)
0153            SCAL1(9)=0.0001
0154            SCAL2(1)=0.                     !BIT COUNTER
0155            SCAL2(17)=1./(2.**14)           !1/2**14
0156            SCAL2(22)=SCAL1(7)
0157            SCAL2(23)=0                     !#N FOR TRANSMITTER
0158            SCAL2(26)=0                     !BETA
0159            SCAL2(29)=0                     !BETA
0160            SCAL3(18)=2.**14                !2**14
0161            SCAL3(20)=SCAL1(7)              !1/2**15
0162            SCAL3(21)=0                     !#N FOR RECEIVER
0163            SCAL3(22)=2047./(16.**3)
0164            SCAL3(23)=-2048./(16.**3)
```

```
0165              INIS(8)=1023                    !BUFFER SIZE - 1
0166              INIS(11)=INIS(8)
0167              INIS(12)=2048                   !STARTING LOCATION FOR RECEIVER BUFFER
0168              A1(1)=AINV                      !INI. VALUE FOR A(1)
0169              A2(1)=AINV
0170              CLOSE(UNIT=4)
                  C
                  C       CREATE INFORMATION FILE
                  C
                  D       WRITE(6,*)'     *****   REAL-TIME PARC ALGORITHM   *****'
                  D       WRITE(6,*)' PARAMETERS FOR THE ADAPTIVE FILTER:'
                  D       WRITE(6,*)' A=',SCAL1(2),'B=',SCAL1(3)
                  D       WRITE(6,*)' TRANSFER BLOCK SIZE =',TBLK
                  D       WRITE(6,*)' FILTERING BLOCK SIZE =',FBLK
                  D       WRITE(6,*)' CORRELATION BLOCK SIZE =',KBLK
                  D       WRITE(6,*)' RUN LENGTH =',SCAL2(5),'BIT # FOR ITS CODE =',BITLNT
                  D       WRITE(6,*)' BIT # FOR LEVEL 1 =',BIT1
                  D       WRITE(6,*)' PITCH PERIOD SEARCHING RANGE =',ITSTRT,'TO',ITEND
                  D       WRITE(6,*)' BIT BLOCK SIZE =',SCAL2(2),'NULL BIT # =',SCAL2(21)
                  D       WRITE(6,*)' GAP SIZE =',IGAP,'PHONY BETA BITS =',SCAL2(6)
                  D       WRITE(6,*)' RMSMIN =',RMSMIN,'ALAD =',ALAD,'ALP =',ALP
                  D       WRITE(6,*)' SMIN =',SMIN,'G =',G,'AINV =',AINV,'STATE =',STATE
                  D       WRITE(6,*)' NUMBER OF QUANTIZER LEVELS =',IK
                  D       WRITE(6,*)' OUTPUT SCALING FACTOR:'
                  D       WRITE(6,*)(OUTSCA(I),I=1,ILL)
                  D       WRITE(6,*)' EXPANSION/CONTRACTION FACTORS:'
                  D       WRITE(6,*)(EXPN(I),I=1,ILL)
                  D       WRITE(6,*)' QUANTIZER THRESHOLDS:'
                  D       WRITE(6,*)(T(I),I=1,IL)
                  D       WRITE(6,*)' # OF BITS CORRESPONDING TO EACH LEVEL:'
                  D       WRITE(6,*)(B(I),I=1,IK)
                  D       WRITE(6,*)' UPPER BETA LIMIT =',UBETA,'LOWER BETA LIMIT =',SCAL1(11)
                  D       WRITE(6,*)' # OF QUANTIZING LEVELS FOR BETA =',IOBETA
                  D       WRITE(6,*)' GAP THRESHOLD =',INIS(1),'FILTER THRESHOLD =',INIS(2)
                  D       WRITE(6,*)' ADAM SAMPLING FREQ =',FREADM,'AOM SAMPLING FREQ =',FREAOM
                  D       WRITE(6,*)' INV. GAIN FACTOR FOR MASKING LSBS =',1./(SCAL3(24)*2.**13)
                  D       WRITE(6,*)' MAX. Q-BUFFER LENGTH =',SCAL3(25)
                  D       WRITE(6,*)' GAIN FACTOR AT ADAM INPUT =',SCAL3(24)*SCAL3(26)
                  C
                  C ----- SET UP ROUTINE FOR IOS TRANSFER
                  C
0171              CALL ASSIGN(3,'SY:IOSDC.CSP')
0172              CALL FPMLD(IOSPGM,SIZE,MSGERR)
0173              IF(MSGERR.EQ.0) GO TO 88
0174              TYPE *,'** ERROR',MSGERR,' IN IOS PROGRAM'
0175              STOP
0176        88    CONTINUE
                  C
                  C ------ SET IOS DATA RATE
                  C
0177              CNTS=256.-(1.536E6/(4.0*2.0*FREADM))
0178              CNTD=256.-(1.536E6/(8.0*1.5*FREADM))
0179              CNTIOS=256.*CNTD+CNTS
0180              IF(CNTIOS.GT.32767.) CNTIOS=CNTIOS-65536.
0181              IOSPGM(2)=CNTIOS
                  C
```

```
        C ----- SET UP HEADER AND TRAILER BLOCK FOR IOS FUNCTION
        C
0182            DO 200 I=1,SIZE
0183            II=SIZE+1-I
0184    200     IOSPGM(II+2)=IOSPGM(II)
        C
0185            IOSPGM(1)=SIZE
0186            IOSPGM(2)=256*BRC+BTR1
        C
0187            IOSPGM(SIZE+2+1)=0
0188            IOSPGM(SIZE+2+2)=2
0189            IOSPGM(SIZE+2+3)=256*0+BTR2
0190            IOSPGM(SIZE+2+4)=0
0191            IOSPGM(SIZE+2+5)=0
0192            IOSPGM(SIZE+2+6)=-1
0193            IOSPGM(SIZE+2+7)=-1
0194            IOSPGM(SIZE+2+8)=-1
0195            IOSPGM(SIZE+2+9)=-1
0196            IOSPGM(SIZE+2+10)=-1
        C
        C
        C       MAP BUFFER COFINGURATION
        C
0197    85      CONTINUE
0198            M=MPOPN(1)
0199            M=MPIBC(0)
0200            M=MPCLB(IBUS3+ADM1,3072.,126.,REAL,CONTIG,SHORT)
0201            M=MPCLB(IBUS3+ADM2,3072.+126.,126.,REAL,CONTIG,SHORT)
0202            M=MPCLB(IBUS2+AOM1,3584.,252.,REAL,CONTIG,SHORT)
0203            M=MPCLB(IBUS2+AOM2,3840.,252.,REAL,CONTIG,SHORT)
0204            M=MPCLB(IBUS2+AOM3,3584.,126.,REAL,CONTIG,SHORT)
0205            M=MPCLB(IBUS2+AOM4,3840.,126.,REAL,CONTIG,SHORT)
0206            M=MPCLB(IBUS3+SAMP,0.,1024.,REAL,CONTIG,SHORT)
0207            M=MPCLB(IBUS2+TXVHAT,0.,1024.,REAL,CONTIG,SHORT)
0208            M=MPCLB(IBUS3+T1SHAT,1024.,1024.,REAL,CONTIG,SHORT)
0209            M=MPCLB(IBUS2+TXPARA,3072.,80.,REAL,CONTIG,LONG)
0210            M=MPCLB(IBUS3+RCVHAT,2048.,1024.,REAL,CONTIG,SHORT)
0211            M=MPCLB(IBUS3+RCVHT2,3068.,4.,REAL,CONTIG,SHORT)
0212            M=MPCLB(IBUS2+R1SHAT,2048.,1024.,REAL,CONTIG,SHORT)
0213            M=MPCLB(IBUS2+RCPARA,4096.,80.,REAL,CONTIG,LONG)
0214            M=MPCLB(IBUS1+IDAOM,23000.,250.,FIXED,CONTIG,LONG)
0215            M=MPCLB(IBUS1+IDADM,23250.,250.,FIXED,CONTIG,LONG)
0216            M=MPCLB(IBUS1+IDIOS,23500.,200.,FIXED,CONTIG,LONG)
        C
0217            M=MPCLB(IBUS1+ENTRB,24000.,100.,FIXED,CONTIG,LONG)
0218            M=MPCLB(IBUS1+ENTRT,23900.,30.,FIXED,CONTIG,LONG)
0219            M=MPCLB(IBUS1+DCRCT,24100.,256.,FIXED,CONTIG,LONG)
0220            M=MPCLB(IBUS1+DCRCB,24400.,128.,FIXED,CONTIG,LONG)
        C
0221            M=MPCLB(IBUS1+QTR1,25000.,600.,FIXED,CONTIG,LONG)
0222            M=MPCLB(IBUS1+BTR1,25700.,200.,FIXED,CONTIG,LONG)
0223            M=MPCLB(IBUS1+QRC1,26000.,600.,FIXED,CONTIG,LONG)
0224            M=MPCLB(IBUS1+QTR2,27000.,600.,FIXED,CONTIG,LONG)
0225            M=MPCLB(IBUS1+BTR2,27700.,200.,FIXED,CONTIG,LONG)
0226            M=MPCLB(IBUS1+QRC2,28000.,600.,FIXED,CONTIG,LONG)
0227            M=MPCLB(IBUS1+BRC,29000.,1024.,FIXED,CONTIG,LONG)
```

```
C
C       INITIALIZE THESE BUFFERS
C
        M=VSMA1(ADM1,0,ADM1,0)
        M=VSMA1(ADM2,0,ADM2,0)
        M=VSMA1(AOM1,0,AOM1,0)
        M=VSMA1(AOM2,0,AOM2,0)
        M=VSMA1(AOM3,0,AOM3,0)
        M=VSMA1(AOM4,0,AOM4,0)
        M=VSMA1(SAMP,0,SAMP,0)
        M=VSMA1(TXVHAT,0,TXVHAT,0)
        M=VSMA1(T1SHAT,0,T1SHAT,0)
        M=VSMA1(RCVHAT,0,RCVHAT,0)
        M=VSMA1(R1SHAT,0,R1SHAT,0)
C
C ----- INITIALIZE Q-BUFFERS FOR ENCODER
C
        DO 90 I=1,130
90      DECT(I)=2
        DECT(1)=1
        DECT(128)=-12
        DECT(129)=-12
        DECT(130)=-12
        M=MPWDB(QTR1,DECT(1),2,0,DECT(130))
        M=MPWDB(QTR2,DECT(1),2,0,DECT(130))
C
C----- CONSTRUCT DECODE TABLES
C
C       1.DECT - SIZE:256
C
        DO 100 I=1,256
        DECT(I)=0
100     CONTINUE
C
        DO 102 I=1,14
        II=DECTAB(1,I)+1
        DECT(II)=DECTAB(2,I)*2
102     CONTINUE
C
C       2.DECB - SIZE:128
C
        DO 105 I=1,128
        DECB(I)=48
105     CONTINUE
C
        DO 106 I=1,97
        II=ENCB(I)+1
        DECB(II)=I-1
106     CONTINUE
C
        DO 107 I=1,8
        II=PHBTA(I)+1
        DECB(II)=-1
107     CONTINUE
C
C ----- INITIALIZE FUNCTION LIST SELECTOR
```

```
        C
0265            ISEL(1)=0
0266            ISEL(2)=-1
0267            ISEL(3)=56              ! SAME AS ARG OF TX UPDATE
        C
        C ----- OFFSET
        C
0268            OFFSET=2
        C
        C
        C
        C ----- INITIALIZE ENCODE-DECODE TABLES
        C
0269            M=MPWDB(ENTRT,ENCT(1),2,0,ENCT(300))
0270            M=MPWDB(ENTRB,ENCB(1),2,0,ENCB(97))
0271            M=MPWDB(DCRCT,DECT(1),2,0,DECT(256))
0272            M=MPWDB(DCRCB,DECB(1),2,0,DECB(128))
        C       INITIALIZE PARAMETERS, SCALARS FOR PARC
        C
0273            M=MPWST(50,SCAL1(1),15,1)
0274            M=MPWST(65,SCAL2(1),29,1)
0275            M=MPWST(94,SCAL3(1),26,1)
0276            M=MPIWS(50,INIS(1),14,1)
0277            M=MPWDB(TXPARA,A1(1),4,1,A1(80))
0278            M=MPIWS(IFSEL,ISEL(1),3,0)
0279            M=MPWDB(RCPARA,A2(1),4,1,A2(80))
        C
        C ----- SET UP IOS TRANSFER ROUTINE AND LOAD IOS
        C
0280            M=MPWDB(IDIOS,IOSPGM(1),2,0,IOSPGM(200))
0281            M=MPLDS(IOS,IOS2,IDIOS)
        C
        C ----- LOAD ADAM, AOM FUNCTIONS
        C
0282            M=ADMSD(IDADM,2.0,1.0,2,CHAN1,ADM2,ADM1)
0283            M=AOMID(IDAOM,1.0,2,AOM2,AOM1,OFFSET)
0284            M=MPLDS(ADM,IOS2,IDADM)
0285            M=MPLDS(AOM,IOS2,IDAOM)
        C
        C       FUNCTION LISTS
        C
        C ----- FUNCTION LIST FOR Q-LEVEL LOOP-BACK OPERATION
        C
0286            M=MPBFL(IFPRC)
0287            M=MPWT(2,ADM1)
0288            M=PICH3(ITBTR1,ADM1)
0289            M=PCTX(ITBTR1+2,QTR1)
0290            M=MPWT(1,1)
0291            M=RC(ITBTR1,59)                 !88 ---> 94
0292            M=TX(56)                        !SAME AS !SEL(3)
0293            M=PCRC(ITBTR1+1,QTR1,AOM3)              !89 ---> 95
0294            M=MPWT(2,ADM2)
0295            M=PICH3(ITBTR2,ADM2)
0296            M=PCTX(ITBTR2+2,QTR2)
0297            M=MPWT(1,1)
0298            M=RC(ITBTR2,59)                 !91 ---> 96
```

```
0299            M=TX(56)                        !SAME AS ISEL(3)
0300            M=PCRC(ITBTR2+1,QTR2,AOM4)              !92 ---> 97
0301            M=MPEFL(0)
      C
      C ----- MAIN FUNCTION LIST FOR PARC THRU IOS
      C
0302            M=MPBFL(IFPIOS)
0303            M=MPRNS(IOS,IOS2,0)
0304            M=INI(50,NRUNL)
0305            M=MPRNS(IOS,IOS2,4)
0306            M=MPRAA(ADM,0,AOM,0)
0307            M=MPWT(2,ADM2)
0308            M=MPWHL(0,0,IFMODE)
0309            M=MPEFL(0)
      C
      C ----- FUNC LIST TO SELECT OPERATION MODE
      C
0310            M=MPBFL(IFMODE)
0311            M=VSMA1(AOM1,0,AOM1,0)
0312            M=VSMA1(AOM2,0,AOM2,0)
0313            M=VSMA1(R1SHAT,0,R1SHAT,0)
0314            M=VSMA1(RCVHT2,0,RCVHT2,0)
0315            M=MPIWL(IFSEL,1,1,IFSYNC)
0316            M=RCINI(61)
0317            M=MPIWL(IFSEL,0,1,IFRCVR)
0318            M=MPEFL(0)
      C
      C ----- SYNC FUNC. LIST - MAIN
      C
0319            M=MPBFL(IFSYNC)
0320            M=MPIIF(IFSEL+1,1,1,IFSYN1,IFSYN2)
0321            M=MPEFL(0)
      C
      C ----- RECEIVER FUNC. LIST - MAIN
      C
0322            M=MPBFL(IFRCVR)
0323            M=MPIIF(IFSEL+1,1,1,IFREC1,IFREC2)
0324            M=MPEFL(0)
      C
      C ----- SYNCHRO FUNC LIST - 1
      C
0325            M=MPBFL(IFSYN1)
0326            M=MPWT(2,ADM1)
0327            M=ENCD(QTR2,BTR2,ENTRB,ENTRT,ITBTR2,ITBRC1,59)
0328            M=PICH3(ITBTR1,ADM1)
0329            M=PCTX(ITBTR1+2,QTR1)
0330            M=DECD(BRC,IFSEL,QRC1,ITBRC1,QRC2,ITBRC2,DCRCT,DCRCB,IGAP)
0331            M=MPWT(1,1)
0332            M=TX(56)
0333            M=MPEFL(0)
      C
      C ----- SYNCHRO FUNC LIST - 2
      C
0334            M=MPBFL(IFSYN2)
0335            M=MPWT(2,ADM2)
0336            M=ENCD(QTR1,BTR1,ENTRB,ENTRT,ITBTR1,ITBRC2,59)
```

```
0337            M=PICH3(ITBTR2,ADM2)
0338            M=PCTX(ITBTR2+2,QTR2)
0339            M=DECD(BRC,IFSEL,QRC1,ITBRC1,QRC2,ITBRC2,DCRCT,DCRCB,IGAP)
0340            M=MPWT(1,1)
0341            M=TX(56)
0342            M=MPEFL(0)
        C
        C ----- PARC FULL OP FUNC LIST - 1
        C
0343            M=MPBFL(IFREC1)
0344            M=MPWT(2,ADM1)
0345            M=RC(ITBRC1,59)
0346            M=PICH3(ITBTR1,ADM1)
0347            M=ENCD(QTR2,BTR2,ENTRB,ENTRT,ITBTR2,ITBRC1,59)
0348            M=PCTX(ITBTR1+2,QTR1)
0349            M=PCRC(ITBRC1+1,QRC1,AOM3)
0350            M=DECD(BRC,IFSEL,QRC1,ITBRC1,QRC2,ITBRC2,DCRCT,DCRCB,IGAP)
0351            M=MPWT(1,1)
0352            M=TX(56)
0353            M=MPEFL(0)
        C
        C ----- PARC FULL OP FUNC LIST - 2
        C
0354            M=MPBFL(IFREC2)
0355            M=MPWT(2,ADM2)
0356            M=RC(ITBRC2,59)
0357            M=PICH3(ITBTR2,ADM2)
0358            M=ENCD(QTR1,BTR1,ENTRB,ENTRT,ITBTR1,ITBRC2,59)
0359            M=PCTX(ITBTR2+2,QTR2)
0360            M=PCRC(ITBRC2+1,QRC2,AOM4)
0361            M=DECD(BRC,IFSEL,QRC1,ITBRC1,QRC2,ITBRC2,DCRCT,DCRCB,IGAP)
0362            M=MPWT(1,1)
0363            M=TX(56)
0364            M=MPEFL(0)
        C
        C ----- FUNCTION LIST TO RUN PARC WITH Q-LEVEL LOOP-BACK
        C
0365            M=MPBFL(IFPQL)
0366            M=INI(50,NRUNL)
0367            M=MPRNS(IOS,IOS2,0)
0368            M=MPRAA(ADM,0,AOM,0)
0369            M=MPWT(2,ADM2)
0370            M=RCINI(61)
0371            M=MPWHL(0,0,IFPRC)
0372            M=MPEFL(0)
        C
        C ----- READY TO RUN. SELECT MODE AND GO
        C
0373    400     CONTINUE
0374            TYPE *,'** SELECT OPERATION MODE:'
0375            TYPE *,'    0 - STOP PROGRAM.'
0376            TYPE *,'    1 - Q-LEVEL LOOP-BACK IN MEMORY. (NO IOS)'
0377            TYPE *,'    2 - BIT STREAM LOOP-BACK THRU IOS. (FULL DUP OP.)'
0378            ACCEPT *,LMODE
0379            GO TO(750,500,600) LMODE+1
        C
```

```
FORTRAN IV-PLUS V02-51          12:09:54    02-MAY-80           PAGE 11
PARC96.FTN      /TR:BLOCKS/WR

        C
        C ----- OPERATION MODE 1
        C
#380    500     M=MPXFL(IFPQL)
#381    510     TYPE *,'** PARC IN Q-LEVEL LOOP-BACK MODE  (NO IOS)'
#382            TYPE *,' '
#383            GO TO 700
        C
        C ----- OPERATION MODE 2
        C
#384    600     M=MPXFL(IFPIOS)
#385    610     TYPE *,'** PARC IN FULL DUPLEX MODE (THRU IOS)'
#386            TYPE *,' '
        C
        C ----- STOP ADM,AOM,IOS, CLOSE MAP, AND STOP.
        C
#387    700     CONTINUE
#388            PAUSE '   TYPE RES... TO CONTINUE'
#389            M=MPSAA(AOM)
#390            M=MPSAA(ADM)
#391            M=MPCLS(0)
#392            GO TO 85
#393    750     CONTINUE
#394            M=MPCLS(0)
#395            STOP
#396            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	012752	2805	RW,I,CON,LCL
2	$PDATA	001030	268	RW,D,CON,LCL
3	$IDATA	002024	522	RW,D,CON,LCL
4	$VARS	005756	1527	RW,D,CON,LCL
5	$TEMPS	000004	2	RW,D,CON,LCL

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ADM	I*2	4-005440	ADM1	I*2	4-002334	ADM2	I*2	4-002336	AINV	R*4	4-005664	ALAD	R*4	4-005644
ALP	R*4	4-005650	ALT	I*2	4-005534	AOM	I*2	4-005442	AOM1	I*2	4-002410	AOM2	I*2	4-002412
AOM3	I*2	4-002414	AOM4	I*2	4-002416	BITLNT	R*4	4-005610	BIT1	R*4	4-005604	BOUT1	I*2	4-002430
BOUT2	I*2	4-002432	BRC	I*2	4-005426	BTR1	I*2	4-005416	BTR2	I*2	4-005420	CMPLX	I*2	4-005522
CNTD	R*4	4-005740	CNTIOS	R*4	4-005744	CNTS	R*4	4-005734	CONTIG	I*2	4-005532	DCRCB	I*2	4-005410
DCRCT	I*2	4-005406	ENTRB	I*2	4-005402	ENTRT	I*2	4-005404	FBLK	I*2	4-002436	FFBLK	R*4	4-005570
FIXED	I*2	4-005524	FREADM	R*4	4-005726	G	R*4	4-005660	GAP	R*4	4-005634	HOBETA	R*4	4-002330
I	I*2	4-005706	IBUS1	I*2	4-005536	IBUS2	I*2	4-005540	IBUS3	I*2	4-005542	IDADM	I*2	4-005450
IDAOM	I*2	4-005452	IDIOS	I*2	4-005454	IFMODE	I*2	4-005552	IFPIOS	I*2	4-005550	IFPOL	I*2	4-005544
IFPRC	I*2	4-005546	IFRCVR	I*2	4-005556	IFREC1	I*2	4-005564	IFREC2	I*2	4-005566	IFSEL	I*2	4-005456
IFSYNC	I*2	4-005554	IFSYN1	I*2	4-005560	IFSYN2	I*2	4-005562	IGAP	I*2	4-005632	II	I*2	4-005750
IK	I*2	4-005676	IL	I*2	4-005700	ILL	I*2	4-005702	IOS	I*2	4-005444	IOS2	I*2	4-005446
IQBETA	I*2	4-005724	ITBRC1	I*2	4-005434	ITBRC2	I*2	4-005436	ITBTR1	I*2	4-005430	ITBTR2	I*2	4-005432
ITEND	I*2	4-005620	ITSTRT	I*2	4-005616	J	I*2	4-005704	K8LK	I*2	4-002406	K1	I*2	4-005710
K11	I*2	4-005712	LMODE	I*2	4-005754	LONG	I*2	4-005526	M	I*2	4-005752	MSGERR	I*2	4-005732
N	I*2	4-005674	NRUNL	I*2	4-005614	OFFSET	I*2	4-005400	QBETA	R*4	4-005720	QRC1	I*2	4-005422
QRC2	I*2	4-005424	QTR1	I*2	4-005412	QTR2	I*2	4-005414	RCPARA	I*2	4-002426	RCVHAT	I*2	4-002420
RCVHT2	I*2	4-002440	REAL	I*2	4-005520	RKBLK	R*4	4-005574	RMSMIN	R*4	4-005640	R1SHAT	I*2	4-002422
R2SHAT	I*2	4-002424	SAMP	I*2	4-002340	SHORT	I*2	4-005530	SIZE	I*2	4-003262	SMIN	R*4	4-005654
STATE	R*4	4-005670	TBLK	I*2	4-002434	TEND	R*4	4-005626	TSTRT	R*4	4-005622	TTBLK	R*4	4-005600
TXPARA	I*2	4-002350	TXVHAT	I*2	4-002342	T1SHAT	I*2	4-002344	T2SHAT	I*2	4-002346	UBETA	R*4	4-005714

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
A1	R*4	4-004430	005000	160	(80)
A2	R*4	4-001130	005000	160	(80)
B	R*4	4-002210	000120	40	(20)
CHAN1	I*2	4-005460	000040	16	(16)
DECB	I*2	4-004662	004000	128	(128)
DECT	I*2	4-003662	001000	256	(256)
DECTAB	I*2	4-005302	000070	28	(2,14)
ENCB	I*2	4-003360	000302	97	(97)
ENCT	I*2	4-003264	000074	30	(30)
EXPN	R*4	4-001750	000120	40	(20)
INIS	I*2	4-002352	000034	14	(14)
IOSPGM	I*2	4-002442	000620	200	(200)
ISEL	I*2	4-005372	000006	3	(3)
OUTSCA	R*4	4-001630	000120	40	(20)

```
TRRC.FTN          /TR:BLOCKS/WR

          C
          C       *** HOST SUPPORT MODULES FOR PARC ***
          C
          C       THERE ARE SIX HOST SUPPORT MODULES FOR PARC AS FOLLOWS:
          C
          C       INI(ISA,LRUN) --- INITIALIZATION PROGRAM FOR ALL APS MODULES.
          C       TX(ISA)  --- UPDATE PROGRAM FOR THE APS MODULES OF PARC-TRANSMITTER
          C       RC(SA,ISA) --- UPDATE PROGRAM FOR THE APS MODULES OF PARC-RECEIVER
          C       PICH3(SA,U) --- FOR BETA, T CALCULATION AND FILTER, GAPPING OPERATIONS.
          C       PCTX(SA,U) --- PARC-TRANSMITTER
          C       PCRC(SA,U,V) --- PARC-RECEIVER
          C       RCINI(ISA) --- INITIALIZATION PROGRAM FOR PARC-RECEIVER
          C
          C
0001              INTEGER FUNCTION INI(ISA,LRUN)
          C
          C I.    INI(ISA,LRUN) --- INITIALIZATION PROGRAM FOR ALL APS MODULES.
          C
          C       FCB = 115
          C
          C       CALL FORMAT:
          C
          C       MAP=INI(ISA,LRUN)
          C       WHERE  ISA --- THE STARTING INTEGER SCALAR LOCATION
          C             LRUN --- RUN LENGTH FOR ENCODER
          C
0002              INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
0003              INTEGER ISA,LRUN
0004              COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
          C
0005              INI=0
0006              DO 100 I=1,11
0007    100       FCBRG(I)=0
0008              FCBRG(2)=115
          C       CHECK ARGUMENT
0009              IF(ISA .LT. 0 .OR. ISA .GT. 127)INI=-1
          C
0010              IF(INI .EQ. 0)GO TO 200
0011              CALL MPERR(INI)
0012              RETURN
          C
0013    200       FCBRG(3)=ISA
0014              FCBRG(5)=LRUN
          C
0015              INI=RUNMP(FCBRG(1),FCBSZ(1,4))
          C
0016              RETURN
0017              END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000222	73	RW,I,CON,LCL
3	$IDATA	000014	6	RW,D,CON,LCL
4	$VARS	000006	3	RW,D,CON,LCL
6	MPZZZ	000312	101	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
INI	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HRI	R*4	6-000304	HWS	I*2	4-000000	I	I*2	4-000004	ISA	I*2	F-000002*
LEVEL	I*2	6-000310	LRUN	I*2	F-000004*									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FCB	I*2	6-000260	000014	6	(6)
FCBRG	I*2	6-000000	000026	11	(11)
FCBSZ	I*2	6-000026	000232	77	(11,7)
MPDCB	I*2	6-000274	000010	4	(4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000144						

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000556 183

```
FORTRAN IV-PLUS V02-51          12:47:28    15-APR-80          PAGE 3
TRRC.FTN        /TR:BLOCKS/WR

        C
        C
0001            INTEGER FUNCTION TX(ISA)
        C
        C II.   TX(ISA) --- UPDATE PROGRAM FOR APS MODULES FOR PARC-TRANSMITTER
        C
        C       FCB = 114
        C
        C       CALL FORMAT:
        C
        C       MAP=TX(ISA)
        C       WHERE  ISA --- THE STARTING INTEGER SCALAR LOCATION
        C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
0003            INTEGER ISA
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
        C
0005            TX=0
0006            DO 100 I=1,11
0007    100     FCBRG(I)=0
0008            FCBRG(2)=114
        C       CHECK ARGUMENT
0009            IF(ISA .LT. 0 .OR. ISA .GT. 127)TX=-1
        C
0010            IF(TX .EQ. 0)GO TO 200
0011            CALL MPERR(TX)
0012            RETURN
        C
0013    200     FCBRG(3)=ISA
        C
0014            TX=RUNMP(FCBRG(1),FCBSZ(1,3))
        C
0015            RETURN
0016            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	SCODE1	000214	70	RW,I,CON,LCL
3	SIDATA	000014	6	RW,D,CON,LCL
4	SVARS	000006	3	RW,D,CON,LCL
6	MPZZZ	000312	101	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
TX	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HRI	R*4	6-000304	HWS	I*2	4-000000	I	I*2	4-000004	ISA	I*2	F-000002*
LEVEL	I*2	6-000310												

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FCB	I*2	6-000260	000014	6	(6)
FCBRG	I*2	6-000000	000026	11	(11)
FCBSZ	I*2	6-000026	000232	77	(11,7)
MPDCB	I*2	6-000274	000010	4	(4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000144						

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000550 100

```
FORTRAN IV-PLUS V02-51          12:47:34     15-APR-80          PAGE 5
TRRC.FTN        /TR:BLOCKS/WR

        C
        C
0001            INTEGER FUNCTION RC(SA,ISA)
        C
        C III.  RC(SA,ISA) --- UPDATE PROGRAM FOR APS MODULES OF PARC-RECEIVER
        C
        C       FCB = 113
        C
        C       CALL FORMAT:
        C
        C       MAP=RC(SA,ISA)
        C       WHERE   SA --- SCALAR LOCATION FOR T
        C               ISA --- THE STARTING INTEGER SCALAR LOCATION
        C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
0003            INTEGER SA,ISA
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
        C
0005            RC=0
0006            DO 100 I=1,11
0007    100     FCBRG(I)=0
0008            FCBRG(2)=113
        C       CHECK ARGUMENTS
0009            IF(SA .LT. 0 .OR. SA .GT. 127)RC=-1
0010            IF(ISA .LT. 0 .OR. ISA .GT. 127)RC=-2
        C
0011            IF(RC .EQ. 0)GO TO 200
0012            CALL MPERR(RC)
0013            RETURN
        C
0014    200     FCBRG(9)=SA
0015            FCBRG(11)=ISA
0016            RC=RUNMP(FCBRG(1),FCBSZ(1,5))
        C
0017            RETURN
0018            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000246	83	RW,I,CON,LCL
3	$IDATA	000014	6	RW,D,CON,LCL
4	$VARS	000006	3	RW,D,CON,LCL
6	MPZZZ	000312	101	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
RC	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HRI	R*4	6-000304	HWS	I*2	4-000000	I	I*2	4-000004	ISA	I*2	F-000004*
LEVEL	I*2	6-000310	SA	I*2	F-000002*									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FCB	I*2	6-000260	000014	6	(6)
FCBRG	I*2	6-000000	000026	11	(11)
FCBSZ	I*2	6-000026	000232	77	(11,7)
MPDCB	I*2	6-000274	000010	4	(4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000170						

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000602 193

```
FORTRAN IV-PLUS V02-51          12:47:41     15-APR-80          PAGE 7
TRRC.FTN        /TR:BLOCKS/WR

        C
        C
0001            INTEGER FUNCTION PICH3(SA,U)
        C
        C IV.   PICH3 -- FOR BETA, T COMPUTATION AND FILTER, GAPPING OPERATIONS
        C
        C       FCB = 240
        C
        C       CALL FORMAT:
        C       MAP=PICH3(SA,U)
        C
        C       WHERE  SA --- SCALAR LOCATION FOR ENCD. BETA, BETA AND T
        C              U  --- INPUT ADAM BUFFER
        C
0002            INTEGER SA,U
0003            IDUMY=0
0004            PICH3=FCBGN(240,0,SA,U,IDUMY,0,0,0,0,6)
0005            RETURN
0006            END
```

```
FORTRAN IV-PLUS V02-51          12:47:41    15-APR-80          PAGE 8
TRRC.FTN        /TR:BLOCKS/WR

PROGRAM SECTIONS

NUMBER    NAME        SIZE                ATTRIBUTES

  1       $CODE1   000102      33          RW,I,CON,LCL
  2       $PDATA   000014       6          RW,D,CON,LCL
  3       $IDATA   000030      12          RW,D,CON,LCL
  4       $VARS    000002       1          RW,D,CON,LCL


ENTRY POINTS

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 PICH3  I*2   1-000000


VARIABLES

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 IDUMY  I*2   4-000000   SA     I*2   F-000002*  U      I*2   F-000004*


FUNCTIONS AND SUBROUTINES REFERENCED

 FCBGN


TOTAL SPACE ALLOCATED = 000150    52
```

```
FORTRAN IV-PLUS V02-51          12:47:45    15-APR-80          PAGE 9
TRRC.FTN        /TR:BLOCKS/WR

        C
        C
###1            INTEGER FUNCTION PCTX(SA,U)
        C
        C V.    PCTX -- PARC-TRANSMITTER
        C
        C       FCB = 242
        C
        C       CALL FORMAT:
        C       MAP=PCTX(SA,U)
        C
        C       WHERE  SA --- THE SCALAR LOCATION FOR BETA
        C              U  --- OUTPUT QUANTIZING BUFFER
        C
###2            INTEGER SA,U
###3            IDUMY=0
###4            PCTX=FCBGN(242,0,SA,U,IDUMY,0,0,0,0,6)
###5            RETURN
###6            END
```

```
FORTRAN IV-PLUS V02-51          12:47:45    15-APR-80          PAGE 10
TRRC.FTN        /TR:BLOCKS/WR

PROGRAM SECTIONS

NUMBER    NAME        SIZE                ATTRIBUTES

  1       $CODE1   000102    33           RW,I,CON,LCL
  2       $PDATA   000014     6           RW,D,CON,LCL
  3       $IDATA   000030    12           RW,D,CON,LCL
  4       $VARS    000002     1           RW,D,CON,LCL

ENTRY POINTS

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 PCTX   I*2   1-000000

VARIABLES

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 IDUMY  I*2   4-000000   SA     I*2   F-000002*  U      I*2   F-000004*

FUNCTIONS AND SUBROUTINES REFERENCED

 FCBGN

TOTAL SPACE ALLOCATED = 00150    52
```

```
FORTRAN IV-PLUS V02-51          12:47:48    15-APR-80           PAGE 11
TRRC.FTN        /TR:BLOCKS/WR

        C
        C
0001            INTEGER FUNCTION PCRC(SA,U,V)
        C
        C VI.   PCRC --- PARC-RECEIVER
        C
        C       FCB = 243
        C
        C       CALL FORMAT:
        C       MAP=PCRC(SA,U,V)
        C
        C       WHERE  SA --- SCALAR NUMBER FOR INPUT ENCD. BETA AND T
        C              V  --- AOM BUFFER
        C              U  --- INPUT QUANTIZING BUFFER
        C
0002            INTEGER SA,U,V
0003            PCRC=FCBGN(243,0,SA,U,0,V,0,0,0,7)
0004            RETURN
0005            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE	ATTRIBUTES
1	$CODE1	000134	34	RW,I,CON,LCL
2	$PDATA	000014	6	RW,D,CON,LCL
3	$IDATA	000030	12	RW,D,CON,LCL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
PCRC	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
SA	I*2	F-000002*	U	I*2	F-000004*	V	I*2	F-000006*						

FUNCTIONS AND SUBROUTINES REFERENCED

FCBGN

TOTAL SPACE ALLOCATED = 000150 52

```
FORTRAN IV-PLUS V02-51          12:47:52    15-APR-80           PAGE 13
TRRC.FTN        /TR:BLOCKS/WR

      C
0001            INTEGER FUNCTION RCINI(ISA)
      C
      C VII.  RCINI(ISA) --- INITIALIZATION PROGRAM FOR APS MODULES OF RECEIVER.
      C
      C       FCB = 116
      C
      C       CALL FORMAT:
      C
      C       MAP=RCINI(ISA)
      C       WHERE  ISA --- THE STARTING INTEGER SCALAR LOCATION
      C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
0003            INTEGER ISA
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
      C
0005            RCINI=0
0006            DO 100 I=1,11
0007    100     FCBRG(I)=0
0008            FCBRG(2)=116
      C       CHECK ARGUMENT
0009            IF(ISA .LT. 0 .OR. ISA .GT. 127)RCINI=-1
      C
0010            IF(RCINI .EQ. 0)GO TO 200
0011            CALL MPERR(RCINI)
0012            RETURN
      C
0013    200     FCBRG(3)=ISA
      C
0014            RCINI=RUNMP(FCBRG(1),FCBSZ(1,3))
      C
0015            RETURN
0016            END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	SCODE1	000216	71	RW,I,CON,LCL
3	SIDATA	000014	6	RW,D,CON,LCL
4	SVARS	000006	3	RW,D,CON,LCL
6	MPZZZ	000312	101	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
RCINI	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
HRD	I*2	4-000002	HRI	R*4	6-000304	HWS	I*2	4-000000	I	I*2	4-000004	ISA	I*2	F-000002*
LEVEL	I*2	6-000310												

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FCB	I*2	6-000260	000014	6	(6)
FCBRG	I*2	6-000000	000026	11	(11)
FCBSZ	I*2	6-000026	000232	77	(11,7)
MPDCB	I*2	6-000274	000010	4	(4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	200	1-000146						

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 000552 181

```
FORTRAN IV-PLUS V02-51          12:47:58    15-APR-80          PAGE 15
TRRC.FTN        /TR:BLOCKS/WR

        C
        C

.TRRC/NOSP=TRRC
```

```
        C
0001            INTEGER FUNCTION ENCD(QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID)
        C
        C ------ HOST SUPPORT MODULE FOR SOURCE CODING THE Q-OUTPUT OF
        C       THE PARC ALGORITHM.
        C
        C       FCB=110
        C
        C       CALLING SEQUENCE:
        C       M=ENCD(QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID)
        C
        C       WHERE:
        C       QHAT = Q-OUTPUT BUFFER, LENGTH=400
        C       OUTB = OUTPUT BIT BUFFER, LENGTH=200
        C       BETA = CODING TABLE FOR 99 BETA VALUES, LENGTH=100
        C       ENCT = SOURCE CODE BUFFER, LENGTH=30
        C              CONTAINS FOR EACH LEVEL:
        C                              1ST WORD:CODE LENGTH-1
        C                              2ND WORD:CODE
        C       ISTB = SID IN INTEGER TABLE WHERE T,BETA ARE LOCATED
        C       SID  = SCALAR ID FOR RCVR UPDATE
        C       ISID = INT. SCALAR ID FOR RCVR UPDATE
        C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRD,LEVEL
0003            INTEGER QHAT,OUTB,BETA,ENCT,ISTB,SID,ISID
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
        C
0005            ENCD=0
0006            DO 100 I=1,11
0007            FCBRG(I)=0
0008    100     CONTINUE
0009            FCBRG(2)=110
        C
        C ----- CHECK BID VALUES
        C
0010            IF(QHAT.LT.1.OR.QHAT.GT.63) ENCD=-1
0011            IF(OUTB.LT.1.OR.OUTB.GT.63) ENCD=-2
0012            IF(BETA.LT.1.OR.BETA.GT.63) ENCD=-3
0013            IF(ENCT.LT.1.OR.ENCT.GT.63) ENCD=-4
        C
        C ----- CHECK SCALAR VALUES
        C
0014            IF(ISTB.LT.1.OR.ISTB.GT.127) ENCD=-5
0015            IF(SID.LT.1.OR.SID.GT.127) ENCD=-6
0016            IF(ISID.LT.1.OR.ISID.GT.127) ENCD=-7
        C
0017            IF(ENCD.EQ.0) GO TO 150
0018            CALL MPERR(ENCD)
0019            RETURN
        C
0020    150     CONTINUE
0021            FCBRG(3)=ISTB
0022            FCBRG(4)=OUTB
0023            FCBRG(5)=QHAT
0024            FCBRG(6)=BE A
0025            FCBRG(7)=ENCT
```

```
FORTRAN IV-PLUS V02-51          12:48:04    15-APR-80          PAGE 2
ENCD.FTN        /TR:BLOCKS/WR

00026           FCBRG(9)=SID
00027           FCBRG(11)=ISID
        C
00028           ENCD=RUNMP(FCBRG(1),FCBSZ(1,5))
        C
00029           RETURN
00030           END
```

PROGRAM SECTIONS

NUMBER	NAME	SIZE		ATTRIBUTES
1	$CODE1	000470	156	RW,I,CON,LCL
3	$IDATA	000014	6	RW,D,CON,LCL
4	$VARS	000006	3	RW,D,CON,LCL
6	MPZZZ	000312	101	RW,D,OVR,GBL

ENTRY POINTS

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
ENCD	I*2	1-000000												

VARIABLES

NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS	NAME	TYPE	ADDRESS
BETA	I*2	F-000006*	ENCT	I*2	F-000010*	HRD	I*2	4-000002	HRI	R*4	6-000304	HWS	I*2	4-000000
I	I*2	4-000004	ISID	I*2	F-000016*	ISTB	I*2	F-000012*	LEVEL	I*2	6-000310	OUTB	I*2	F-000004*
QHAT	I*2	F-000002*	SID	I*2	F-000014*									

ARRAYS

NAME	TYPE	ADDRESS	SIZE		DIMENSIONS
FCB	I*2	6-000260	000014	6	(6)
FCBRG	I*2	6-000000	000026	11	(11)
FCBSZ	I*2	6-000026	000232	77	(11,7)
MPDCB	I*2	6-000274	000010	4	(4)

LABELS

LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS	LABEL	ADDRESS
100	**	150	1-000354						

FUNCTIONS AND SUBROUTINES REFERENCED

MPERR RUNMP

TOTAL SPACE ALLOCATED = 001024 266

```
FORTRAN IV-PLUS V02-51          12:48:15     15-APR-80          PAGE 4
ENCD.FTN        /TR:BLOCKS/WR

        C

.ENCD/NOSP=ENCD
```

```
FORTRAN IV-PLUS V02-51          12:48:21    15-APR-80          PAGE 1
TRBUF.FTN       /TR:BLOCKS/WR

        C
0001            INTEGER FUNCTION TRBUF(DBUF1,DBUF2,INIT,CBUF)
        C
        C ----- HOST SUPPORT MODULE FOR DOUBLE BUFFER TO CIRC BUFF TRANSFER
        C       PROGRAM.
        C ----- IOS SIMULATOR IN CSPU. (ON/OFF HOOK SIMULATION)
        C
        C       FCB=112
        C
        C       ARGUMENTS:
        C       DBUF1 = BID OF DOUBLE BUFFER 1
        C       DBUF2 = BID OF DOUBLE BUFFER 2
        C       INIT = INT. SCALAR ID FOR INIT. SELECT SWITCH
        C       INIT+1 = INT. SCALAR ID FOR IOS MODE
        C                0 - OFF HOOK
        C                1 - ON HOOK
        C       CBUF = BID FOR CIRCULAR BUFFER
        C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRI,LEVEL,RUNMP
0003            INTEGER DBUF1,DBUF2,INIT,CBUF
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
        C
0005            TRBUF=0
0006            DO 100 I=1,11
0007            FCBRG(I)=0
0008    100     CONTINUE
0009            FCBRG(2)=112
        C
        C ----- CHECK ARGUMENTS
        C
0010            IF(DBUF1.LT.1.OR.DBUF1.GT.63) TRBUF=-1
0011            IF(DBUF2.LT.1.OR.DBUF2.GT.63) TRBUF=-2
0012            IF(INIT.LT.1.OR.INIT.GT.127) TRBUF=-3
0013            IF(CBUF.LT.1.OR.CBUF.GT.63) TRBUF=-4
        C
0014            IF(TRBUF.EQ.0) GO TO 150
0015            CALL MPERR(TRBUF)
0016            RETURN
        C
0017    150     CONTINUE
0018            FCBRG(3)=INIT
0019            FCBRG(4)=DBUF1
0020            FCBRG(5)=DBUF2
0021            FCBRG(7)=CBUF
        C
0022            TRBUF=RUNMP(FCBRG(1),FCBSZ(1,4))
        C
0023            RETURN
0024            END
```

```
FORTRAN IV-PLUS V02-51          12:48:21    15-APR-80           PAGE 2
TRBUF.FTN       /TR:BLOCKS/WR

PROGRAM SECTIONS

NUMBER    NAME        SIZE                ATTRIBUTES

  1       SCODE1   000326   107            RW,I,CON,LCL
  3       SIDATA   000014     6            RW,D,CON,LCL
  4       SVARS    000004     2            RW,D,CON,LCL
  6       MPZZZ    000310   100            RW,D,OVR,GBL


ENTRY POINTS

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 TRBUF  I*2   1-000000


VARIABLES

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 CBUF   I*2   F-000010*  DBUF1  I*2   F-000002*  DBUF2  I*2   F-000004*  HRI    I*2   6-000304   HWS    I*2   4-000000
 I      I*2   4-000002   INIT   I*2   F-000006*  LEVEL  I*2   6-000306


ARRAYS

 NAME   TYPE  ADDRESS      SIZE       DIMENSIONS

 FCB    I*2   6-000260   000014     6    (6)
 FCBRG  I*2   6-000000   000026    11    (11)
 FCBSZ  I*2   6-000026   000232    77    (11,7)
 MPDCB  I*2   6-000274   000010     4    (4)


LABELS

 LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS

 100       **            150     1-000252


FUNCTIONS AND SUBROUTINES REFERENCED

 MPERR   RUNMP


TOTAL SPACE ALLOCATED = 000656   215

NO FPP INSTRUCTIONS GENERATED
```

```
        C

.TRBUF/NOSP=TRBUF
```

```
FORTRAN IV-PLUS V02-51          12:48:36    15-APR-80          PAGE 1
DECD.FTN        /TR:BLOCKS/WR

        C
0001            INTEGER FUNCTION DECD(ICB,ISL,OQB1,ITB1,OQB2,ITB2,DECT,DECB,IG)
        C
        C----- HOST SUPPORT MODULE FOR DECODING BIT STREAM INPUT FOR THE
        C      PARC RECEIVER.
        C
        C       FCB=111
        C
        C       CALLING SEQUENCE:
        C       M=DECD(ICB,ISL,OQB1,ITB1,OQB2,ITB2,DECT,DECB,IG)
        C
        C       WHERE:
        C       ICB = INPUT BIT BUFFER, CIRCULAR, LENGTH=1024
        C       ISL = FUNCTION LIST SELECTOR
        C               ISL1 = 0  - INITIALIZE,SYNCHRONIZE
        C                     -1  - SYNCHRONIZE
        C                     +1  - DECODE
        C               ISL2 =-1  - FUNC. LIST 1
        C                     +1  - FUNC. LIST 2
        C               ISL3 =INT. SCAL ID FOR XMTR UPDATE
        C       OQB1 = OUTPUT QHAT BUFFER 1, LENGTH=400
        C       ITB1 = INT. SC. TABLE LOC. FOR T,BETA 1, LENGTH=2
        C       OQB2 = OUTPUT QHAT BUFFER 2, LENGTH=400
        C       ITB2 = INT. SC. TABLE LOC. FOR T,BETA 2, LENGTH=2
        C       DECT = Q-LEVEL DECODE TABLE, LENGTH=256
        C       DECB = BETA DECODE TABLE   , LENGTH=128
        C       IG   = GAP SIZE IN INTEGER FORMAT
        C
        C
0002            INTEGER FCBRG,FCBSZ,HWS,FCB,MPDCB,HRI,LEVEL,RUNMP
0003            INTEGER ICB,ISL,OQB1,ITB1,OQB2,ITB2,DECT,DECB
0004            COMMON /MPZZZ/FCBRG(11),FCBSZ(11,7),FCB(6),MPDCB(4),HRI,LEVEL
        C
0005            DECD=0
0006            DO 100 I=1,11
0007            FCBRG(I)=0
0008    100     CONTINUE
0009            FCBRG(2)=111
        C
        C----- CHECK BID VALUES
        C
0010            IF(ICB.LT.1.OR.ICB.GT.63) DECD=-1
0011            IF(OQB1.LT.1.OR.OQB1.GT.63) DECD=-3
0012            IF(OQB2.LT.1.OR.OQB2.GT.63) DECD=-5
0013            IF(DECT.LT.1.OR.DECT.GT.63) DECD=-7
0014            IF(DECB.LT.1.OR.DECB.GT.63) DECD=-8
        C
        C----- CHECK SID VALUES
        C
0015            IF(ISL.LT.1.OR.ISL.GT.127) DECD=-2
0016            IF(ITB1.LT.1.OR.ITB1.GT.127) DECD=-4
0017            IF(ITB2.LT.1.OR.ITB2.GT.127) DECD=-6
        C
        C----- CHECK GAP SIZE
        C
0018            IF(IG.LT.1.OR.IG.GT.125) DECD=-9
```

```
        C
##19            IF(DECD.EQ.#) GO TO 15#
##2#            CALL MPERR(DECD)
##21            RETURN
        C
##22    15#     CONTINUE
##23            FCBRG(3)=ISL
##24            FCBRG(4)=OQB2
##25            FCBRG(5)=OQB1
##26            FCBRG(6)=ITB2
##27            FCBRG(7)=ITB1
##28            FCBRG(8)=DECT
##29            FCBRG(9)=DECB
##3#            FCBRG(1#)=IG
##31            FCBRG(11)=ICB
        C
##32            DECD=RUNMP(FCBRG(1),FCBSZ(1,5))
        C
##33            RETURN
##34            END
```

```
FORTRAN IV-PLUS V02-51          12:48:36     15-APR-80          PAGE 3
DECD.FTN        /TR:BLOCKS/WR

PROGRAM SECTIONS

NUMBER   NAME       SIZE                ATTRIBUTES

  1      $CODE1   000542   177           RW,I,CON,LCL
  3      $IDATA   000014     6           RW,D,CON,LCL
  4      $VARS    000004     2           RW,D,CON,LCL
  6      MPZZZ    000310   100           RW,D,OVR,GBL


ENTRY POINTS

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 DECD   I*2   1-000000


VARIABLES

 NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS    NAME   TYPE  ADDRESS

 DECB   I*2   F-000020*  DECT   I*2   F-000016*  HRI    I*2   6-000304   HWS    I*2   4-000000   I      I*2   4-000002
 ICB    I*2   F-000002*  IG     I*2   F-000022*  ISL    I*2   F-000004*  ITB1   I*2   F-000010*  ITB2   I*2   F-000014*
 LEVEL  I*2   6-000306   OQB1   I*2   F-000006*  OQB2   I*2   F-000012*


ARRAYS

 NAME   TYPE  ADDRESS      SIZE       DIMENSIONS

 FCB    I*2   6-000260   000014     6   (6)
 FCBRG  I*2   6-000000   000026    11   (11)
 FCBSZ  I*2   6-000026   000232    77   (11,7)
 MPDCB  I*2   6-000274   000010     4   (4)


LABELS

 LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS        LABEL   ADDRESS

 100       **            150     1-000430


FUNCTIONS AND SUBROUTINES REFERENCED

 MPERR   RUNMP


TOTAL SPACE ALLOCATED = 001072   285

NO FPP INSTRUCTIONS GENERATED
```

```
        C

.DECD/NOSP=DECD
```

```
(00001) *       MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980
(00002) *        ************************************************************
(00003) *        *                                                          *
(00004) *        *    REAL TIME IMPLEMENTATION OF THE P A R C ALGORITHM      *
(00005) *        *                                                          *
(00006) *        ************************************************************
(00007) *
(00008) *        PROGRAM: PARC.MAP                          MAWLIN YEH
(00009) *                                                   ARVIND S ARORA
(00010) *        JAN. 30, 1980
(00011) *
(00012) *
(00013) *           P A R C  (PITCH EXTRACTED, ADAPTIVE, PREDICTIVE RESIDUAL
(00014) *        ENCODER) IS A SPEECH COMPRESSION ALGORITHM DEVELOPED AT THE
(00015) *        DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF NOTRE
(00016) *        DAME UNDER CONTRACT FROM THE DEFENCE COMMUNICATION AGENCY.
(00017) *        THE ALGORITHM HAS BEEN CURRENTLY DEVELOPED AND OPTIMIZED
(00018) *        FOR DIGITAL TRANSMISSION OF SPEECH AT 9600 BAUD.
(00019) *
(00020) *           THE REAL TIME IMPLEMENTATION OF P A R C HAS BEEN DONE ON THE
(00021) *        MAP-300 ARRAY PROCESSOR. SEVEN OF THE EIGHT PROGRAM MODULES
(00022) *        USED IN THE IMPLEMENTATION ARE CONTAINED IN THIS FILE:
(00023) *          1. INITIALIZATION AND UPDATING.
(00024) *          2. ESTIMATION OF THE PITCH PERIOD AND THE CORRELATION
(00025) *             COEFFICIENT REQUIRED FOR PITCH REDUNDANCY REMOVAL (PRR).
(00026) *          3. PARC-TRANSMITTER.
(00027) *          4. PARC-RECEIVER.
(00028) *          5. DOUBLE-BUFFER (189*2) TO CIRCULAR-BUFFER (1024) TRANSFER
(00029) *             PROGRAM. THIS IS USED TO SIMULATE THE FUNCTION OF THE
(00030) *             IOS2.
(00031) *          6. THE SOURCE ENCODER PREPARES THE DIGITAL BIT STREAM BY
(00032) *             ENCODING THE QUANTIZER LEVELS AND THE PRR SIDE INFORMATION
(00033) *             PUT OUT BY THE PARC TRANSMITTER. IT USES A VARIABLE-LENGTH
(00034) *             TO VARIABLE-LENGTH ENCODING SCHEME.
(00035) *          7. THE DECODER PERFORMS TWO FUNCTIONS. IT FIRST ACQUIRES FRAME
(00036) *             SYNCHRONIZATION ON THE INCOMING BIT STREAM. IT THEN DECODES
(00037) *             THE BIT STREAM TO THE QUANTIZER LEVELS AND THE PRR INFOR-
(00038) *             MATION FOR USE BY THE PARC RECEIVER.
(00039) *
(00040) *
(00041)         EJECT
```

```
                    (00042) *       ********************************************************
                    (00043) *       *                                                      *
                    (00044) *       *      INITIALIZATION AND UPDATE FOR  P A R C          *
                    (00045) *       *                                                      *
                    (00046) *       ********************************************************
                    (00047) *
                    (00048) *       THIS MODULE CONTAINS 4 PROGRAMS: RCS, TXS INIS AND RCINIS
                    (00049) *
                    (00050) *       RCS: TO UPDATE THE ADDRESS POINTERS IN PARC-RECEIVER
                    (00051) *       TXS: TO UPDATE THE ADDRESS POINTERS IN PARC-TRANSMITTER
                    (00052) *       INIS: TO INITIALIZE THE PARAMETERS IN APS MODULES, ENCODER, DECODER
                    (00053) *       RCINIS: TO INITIALIZE THE POINTERS OF RECEIVER
                    (00054) *
                    (00055) *       SYMBOL DEFINITION TO INTERFACE WITH THE MAP-300 EXEC, VER. 3.5:
                    (00056) *
          00000582  (00057)         BCT$BA=$582
          000008C4  (00058)         FDT$UFCB=$8C4
          00000582  (00059)         ISVT$=$502
          00000382  (00060)         SVT$=$382                       ;SCALAR TABLE
          0001FFCE  (00061)         SYS$FLGS=$1FFCE                 ;SYSTEM FLAG POSITION
          000000FF  (00062)         MSK$RBYT=$00FF                  ;MASK OFF LEFT BYTE
          00000001  (00063)         F1$=$0001
          00000583  (00064)         ISVT1$=ISVT$+1
          00000584  (00065)         ISVT2$=ISVT$+2
          00000585  (00066)         ISVT3$=ISVT$+3
          00000586  (00067)         ISVT4$=ISVT$+4
          00000587  (00068)         ISVT5$=ISVT$+5
          00000003  (00069)         WRD4=3
          00000004  (00070)         WRD5=4
                    (00071) *
                    (00072) *       DISPATCH TABLE ENTRIES
                    (00073) *
          0000006E  (00074) FCB = 110
          000008C4  (00075) #L = FDT$UFCB+(FCB-110)*2
    00008C4 00004FC2 (00076)         ADDR    ENCD$
    00008C6 00005162 (00077)         ADDR    DCDR$
    00008C8 00005116 (00078)         ADDR    TR$DC
    00008CA 00004800 (00079)         ADDR    RC$
    00008CC 00004836 (00080)         ADDR    TX$
    00008CE 0000485C (00081)         ADDR    INI$
    00008D0 00004890 (00082)         ADDR    RCINI$
                    (00083) *
          00004800  (00084) #L = $4800
                    (00085) *
```

PAGE 3: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

(00086) EJECT

```
                    (00087) *
                    (00088) *       RCS       MODULE TO UPDATE RECEIVER PARAMETERS --- PITCH PERIOD,
                    (00089) *                 VHAT POINTER, SHAT POINTER, AOM POINTER.
                    (00090) *
                    (00091) *       INTEGER SCALAR SEQUENCE: ISTART, BUFFER SIZE - 1, STARTING LOCATION OF
                    (00092) *       RECEIVER BUFFER, PITCH REPETITION SIZE, TRANSFER SIZE.
                    (00093) *
                    (00094) *       FCB FORMAT  ( 16 BIT WORD FORMAT SHOWN )
                    (00095) *
                    (00096) *
                    (00097) *       WORD      |     LEFT BYTE      |     RIGHT BYTE
                    (00098) *                 |                    |
                    (00099) *       ----------|--------------------|------------------------
                    (00100) *                 |                    |
                    (00101) *       0         |     POINTER TO NEXT FCB AND FUNCTION LIST
                    (00102) *                 |                    |
                    (00103) *       ----------|--------------------|------------------------
                    (00104) *                 |                    |
                    (00105) *       1         |       113          |           0
                    (00106) *                 |                    |
                    (00107) *       ----------|--------------------|------------------------
                    (00108) *                 |                    |
                    (00109) *       2         |        0           |           0
                    (00110) *                 |                    |
                    (00111) *       ----------|--------------------|------------------------
                    (00112) *                 |                    |
                    (00113) *       3         |        0           |           0
                    (00114) *                 |                    |
                    (00115) *       ----------|--------------------|------------------------
                    (00116) *                 |                    |
                    (00117) *       4         |        0           |     SCALAR #
                    (00118) *                 |                    |
                    (00119) *       ----------|--------------------|------------------------
                    (00120) *                 |                    |
                    (00121) *       5         |        0           |   INT. SCALAR #
                    (00122) *                 |                    |
                    (00123) *       ----------|--------------------|------------------------
                    (00124) *
                    (00125) *
04000 F0220003 (00126) RCS      MOVMR   R2,WRD4(R1)             ;GET SCALAR #
04002 3A21     (00127)          LLS     R2,1                    ;
04003 F0340382 (00128)          MOVMR   R3,SVT0(R2)             ;GET PITCH PERIOD
04005 F0220004 (00129)          MOVMR   R2,WRD5(R1)             ;GET INTEGER SCALAR #
04007 F0640502 (00130)          MOVMR   R6,ISVT0(R2)            ;ISTART
```

```
0489 4C3C     (00131)          ADDRR   R3,R6                  ;DECODE T
048A E030F13  (00132)          MOVRM   R3,TSRC                ;OUTPUT PITCH PERIOD
048C F030F2D  (00133)          MOVMR   R3,VHSRC               ;GET OLD POINTER VALUE
048E F05000466 (00134)         MOVMR   R5,NSRC                ;GET N OF RECEIVER
0491 4C3A     (00135)          ADDRR   R3,R5                  ;ADD N
0493 F0064005B3 (00136)        MOVMR   R6,ISVT1$(R2)          ;1023
0497 4A3C     (00137)          ANDRR   R3,R6                  ;ROUNDING
0498 F0074005B4 (00138)        MOVMR   R7,ISVT2$(R2)          ;STARTING LOCATION OF RECEIVER BUFFE
049A 4C3E     (00139)          ADDRR   R3,R7                  ;
049B E030F2D  (00140)          MOVRM   R3,VH$RC               ;OUTPUT VHAT POINTER
049D F030F0F  (00141)          MOVMR   R3,SH$RC               ;GET OLD VALUE SHAT POINTER
049F 4C3A     (00142)          ADDRR   R3,R5                  ;ADD N OF RECEIVER
04A0 F051FFCE (00143)          MOVMR   R5,SYS$FLGS            ;GET SYSTEM FLAG
04A2 5A5A0008 (00144)          ANDKR   R5,R5,$8               ;CHECK G3
04A6 0010027  (00145)          JMP     CONT3$,EQ              ;JUMP IF NO PITCH REPETITION
04A8 F00FFFCE (00146)          MOVLM   $7,SYS$FLGS            ;RESET G3
04AA F054005B5 (00147)         MOVMR   R5,ISVT3$(R2)          ;PITCH REPETITION SIZE
04AE 4C3A     (00148)          ADDRR   R3,R5                  ;ADD PITCH REPETITION SIZE
04AF 4A3C     (00149) CONT3$   ANDRR   R3,R6                  ;ROUNDING
04B0 4C3E     (00150)          ADDRR   R3,R7                  ;
04B1 E030F0F  (00151)          MOVRM   R3,SH$RC               ;OUTPUT SHAT POINTER
04B3 F054005B6 (00152)         MOVMR   R5,ISVT4$(R2)          ;GET TRANSFER SIZE
04B7 F030ED3  (00153)          MOVMR   R3,DA$RC               ;GET OLD VALUE
04B9 4C3A     (00154)          ADDRR   R3,R5                  ;ADD TRANSFER SIZE
04BB 4A3C     (00155)          ANDRR   R3,R6                  ;ROUDING
04BC 4C3E     (00156)          ADDRR   R3,R7
04BE E030ED3  (00157)          MOVRM   R3,DA$RC               ;OUTPUT AOM POINTER
04C0 0E70     (00158)          RETURN
04C1 0800     (00159)          EVEN
              (00160) *
              (00161)          EJECT
```

```
                    (00162) *
                    (00163) *       TXS       MODULE TO UPDATE APS PARAMETERS OF  P A R C  TRANSMITTER
                    (00164) *
                    (00165) *       IT UPDATES THE FOLLOWING PARAMETERS:
                    (00166) *
                    (00167) *       DASTX, VHSTX, SSTX
                    (00168) *
                    (00169) *       INTEGER SCALAR SEQUENCE: TBLK, BUFFER SIZE - 1, REPETITION SIZE
                    (00170) *
                    (00171) *                |                    |
                    (00172) *       WORD     |      LEFT BYTE     |     RIGHT BYTE
                    (00173) *                |                    |
                    (00174) *       ---------|--------------------|----------------------
                    (00175) *                |                    |
                    (00176) *       0        |      POINTER TO NEXT FCB AND FUNCTION LIST
                    (00177) *                |                    |
                    (00178) *       ---------|--------------------|----------------------
                    (00179) *                |                    |
                    (00180) *       1        |      114           |     INTEGER SCALAR 0
                    (00181) *                |                    |
                    (00182) *       ---------|--------------------|----------------------
                    (00183) *                |                    |
                    (00184) *       2        |       0            |          0
                    (00185) *                |                    |
                    (00186) *       ---------|--------------------|----------------------
                    (00187) *                |                    |
                    (00188) *       3        |       0            |          0
                    (00189) *                |                    |
                    (00190) *       ---------|--------------------|----------------------
                    (00191) *                |                    |
                    (00192) *       4        |       0            |          0
                    (00193) *                |                    |
                    (00194) *       ---------|--------------------|----------------------
                    (00195) *                |                    |
                    (00196) *       5        |       0            |          0
                    (00197) *                |                    |
                    (00198) *       ---------|--------------------|----------------------
                    (00199) *
                    (00200) *
0483G 7022FF (00201) TXS    MOVWK   R2,R1,MSKSRBYT          ;GET INTEGER ADDRESS
0483I F030430 (00202)       MOVMR   R3,NSTX                 ;R3=0 OF SAMPLES
0483A F0645052 (00203)      MOVMR   R6,ISVTS(R2)            ;R6=TRANSFER BLOCK SIZE
0483C F04A21 (00204)        MOVMR   R4,DASTX                ;
0483E 4C4C (00205)          ADDRR   R4,R6                   ;ADD TRANSFER BLOCK SIZE
```

```
0483F F054050A3 (00206)          MOVMR    R5,ISVT1S(R2)              ;MASK FOR SAMPLE BUFFER
04841 4A4A      (00207)          ANDRR    R4,R5                      ;
04842 E040A421  (00208)          MOVRM    R4,DASTX                   ;OUTPUT TRANSFERING POINTER
04844 F040A425  (00209)          MOVMR    R4,VHSTX
04846 4C46      (00210)          ADDRR    R4,R3                      ;ADD N OF TX
04847 4A4A      (00211)          ANDRR    R4,R5                      ;MASKING
04848 E040A425  (00212)          MOVRM    R4,VHSTX                   ;OUTPUT VHAT POINTER
0484A F040A41D  (00213)          MOVMR    R4,SSTX
0484C 4C46      (00214)          ADDRR    R4,R3                      ;ADD N OF TX
0484D F061FFCE  (00215)          MOVMR    R6,SYSSFLGS                ;GET SYSTEM FLAG
0484F 5A6C0004  (00216)          ANDKR    R6,R6,S4                   ;CHECK G2
04851 8010485B  (00217)          JMP      CONT2S,EQ                  ;JUMP IF G2=0
04853 F800FFCE  (00218)          MOVLM    S6,SYSSFLGS                ;RESET G2
04855 F064050A4 (00219)          MOVMR    R6,ISVT2S(R2)              ;GET PITCH REPETITION SIZE
04857 4C4C      (00220)          ADDRR    R4,R6                      ;ADD PITCH REPETITION SIZE
04858 4A4A      (00221) CONT2S   ANDRR    R4,R5                      ;MASKING
04859 E040A41D  (00222)          MOVRM    R4,SSTX                    ;OUTPUT S POINTER
0485B 0E700     (00223)          RETURN
                (00224)          EVEN
                (00225) *
                (00226)          EJECT
```

```
(00227) *
(00228) *       INIS      MODULE TO INITIALIZE APS PARAMETERS.
(00229) *
(00230) *       IT INITIALIZES THE FOLLOWING PARAMETERS:
(00231) *
(00232) *       PITCH REPETITION THRESHOLD, FILTER THRESHOLD, FILTERING SIZE, ISTART,
(00233) *       CORRELATION SIZE, PITCH REPETITION SIZE, NSTX, SSTX, DASTX, VHSTX
(00234) *
(00235) *       INTEGER SCALAR SEQUENCE: PITCH REPETITION THRESHOLD, FILTER THRESHOLD,
(00236) *       FILTER SIZE - 1, ISTART - 1, KBLK - 1, PITCH REPETITION SIZE
(00237) *
(00238) *       FCB FORMAT   ( 16 BIT WORD FORMAT SHOWN )
(00239) *
(00240) *                    |                    |
(00241) *       WORD         |     LEFT BYTE      |      RIGHT BYTE
(00242) *                    |                    |
(00243) *       -------------|--------------------|----------------------
(00244) *                    |                    |
(00245) *       0            |     POINTER TO NEXT FCB AND FUNCTION LIST
(00246) *                    |                    |
(00247) *       -------------|--------------------|----------------------
(00248) *                    |                    |
(00249) *       1            |       115          |      INTEGER SCALAR #
(00250) *                    |                    |
(00251) *       -------------|--------------------|----------------------
(00252) *                    |                    |
(00253) *       2            |        0           |        0
(00254) *                    |                    |
(00255) *       -------------|--------------------|----------------------
(00256) *                    |                    |
(00257) *       3            |        0           |        0
(00258) *                    |                    |
(00259) *       -------------|--------------------|----------------------
(00260) *                    |                    |
(00261) *       4            |        0           |        0
(00262) *                    |                    |
(00263) *       -------------|--------------------|----------------------
(00264) *                    |                    |
(00265) *       5            |        0           |        0
(00266) *                    |                    |
(00267) *       -------------|--------------------|----------------------
(00268) *
..03FF (00269) INIS    MOVWK    R2,R1,MSKSRBYT              ;GET INTEGER SCALAR #
:2502  (00270)         MOVMR    R3,ISVTS(R2)                ;PITCH REPETITION THRESHOLD
```

```
0486 E030A4F (0271)         MOVRM   R3,INSGTH                   ;WRITE PITCH REPETITION THRESHOLD
0462 F034050 (0272)         MOVMR   R3,ISVT1$(R2)               ;FILTER THRESHOLD
0464 E030A57 (0273)         MOVRM   R3,INSFTH                   ;WRITE FILTER THRESHOLD
0466 F034054 (0274)         MOVMR   R3,ISVT2$(R2)               ;FILTER SIZE - 1
0468 E030A6B (0275)         MOVRM   R3,INSFBS                   ;WRITE FILTER SIZE - 1
046A F034055 (0276)         MOVMR   R3,ISVT3$(R2)               ;STARTING CORRELATION # - 1
046C E030A8F (0277)         MOVRM   R3,INSCST                   ;WRITE STARTING CORRELATION #
046E F034056 (0278)         MOVMR   R3,ISVT4$(R2)               ;CORRELATION BLOCK SIZE - 1
0470 E030A9F (0279)         MOVRM   R3,INSCSZ                   ;WRITE CORRELATION BLOCK SIZE - 1
0472 F034057 (0280)         MOVMR   R3,ISVT5$(R2)               ;PITCH REPETITION SIZE
0474 E030AF1 (0281)         MOVRM   R3,INSGSZ                   ;WRITE PITCH REPETITION SIZE
0476 E030CB1 (0282)         MOVRM   R3,INSTX                    ;
0478 2731    (0283)         DECR    R3,1                        ;PITCH REPETITION SIZE - 1
0479 E030F1B (0284)         MOVRM   R3,INSRC                    ;
047B 4E36    (0285)         SUBRR   R3,R3                       ;RESET R3
047C E030043 (0286)         MOVRM   R3,NSTX                     ;RESET
047E E030A21 (0287)         MOVRM   R3,DASTX
0480 E030A25 (0288)         MOVRM   R3,VHSTX
0482 E030A1D (0289)         MOVRM   R3,SSTX
             (0290) ;
             (0291) ;  --- INITIALIZATION FOR THE ENCODER AND DECODER
             (0292) ;
0484 2611    (0293)         INCR    R1,1                        ; POINT TO RUN LENGTH
0485 7022FF  (0294)         MOVWK   R2,R1,SFF                   ;VALUE OF RUN LENGTH
0487 E025115 (0295)         MOVRM   R2,LSRUN                    ;INIT VAL FOR ENCD, DECD
0489 0C20    (0296)         CCR     R2                          ;R2=-1
048A E025114 (0297)         MOVRM   R2,VSDEL                    ;BIT CNT.=-1, FOR ENCD
048C 0D20    (0298)         CLR     R2                          ;R2=0
048D E0254B8 (0299)         MOVRM   R2,PSBASE                   ;BASE OF FRAME POINTER,DEC
             (0300) ;
048F 0E70    (0301)         RETURN
             (0302)         EVEN
             (0303) *
             (0304)         EJECT
```

```
                    (00305) *
                    (00306) *       RCINIS    MODULE TO INITIALIZE APS PARAMETERS OF RECEIVER.
                    (00307) *
                    (00308) *       IT INITIALIZES THE FOLLOWING PARAMETERS:
                    (00309) *
                    (00310) *       NSRC, VHSRC, SHSRC, DASRC AND G3
                    (00311) *
                    (00312) *       INTEGER SCALAR SEQUENCE: STARTING LOCATION OF RECEIVER BUFFER
                    (00313) *       TRANSFER BLOCK SIZE
                    (00314) *
                    (00315) *       FCB FORMAT   ( 16 BIT WORD FORMAT SHOWN )
                    (00316) *
                    (00317) *                    |                    |
                    (00318) *       WORD         |     LEFT BYTE      |      RIGHT BYTE
                    (00319) *                    |                    |
                    (00320) *       -------------|--------------------|----------------------
                    (00321) *                    |                    |
                    (00322) *       0            |     POINTER TO NEXT FCB AND FUNCTION LIST
                    (00323) *                    |                    |
                    (00324) *       -------------|--------------------|----------------------
                    (00325) *                    |                    |
                    (00326) *       1            |       116          |      INTEGER SCALAR #
                    (00327) *                    |                    |
                    (00328) *       -------------|--------------------|----------------------
                    (00329) *                    |                    |
                    (00330) *       2            |        0           |        0
                    (00331) *                    |                    |
                    (00332) *       -------------|--------------------|----------------------
                    (00333) *                    |                    |
                    (00334) *       3            |        0           |        0
                    (00335) *                    |                    |
                    (00336) *       -------------|--------------------|----------------------
                    (00337) *                    |                    |
                    (00338) *       4            |        0           |        0
                    (00339) *                    |                    |
                    (00340) *       -------------|--------------------|----------------------
                    (00341) *                    |                    |
                    (00342) *       5            |        0           |        0
                    (00343) *                    |                    |
                    (00344) *       -------------|--------------------|----------------------
                    (00345) *
4898 7822FFFF (00346) RCINIS  MOVWK   R2,R1,MSKSRBYT          ;GET INTEGER SCALAR #
4892 F034052 (00347)          MOVMR   R3,ISVT$(R2)            ;STARTING LOCATION OF RECEIVER BUF
4894 E034F2D (00348)          MOVRM   R3,VHSRC                ;REAET
```

```
#4896 E#3#4F#F (##349)          MOVRM   R3,SH$RC                    ;
#4898 F#44#5#4 (##35#)          MOVMR   R4,ISVT2$(R2)               ;TRANSFER SIZE
#489A 4E38     (##351)          SUBRR   R3,R4
#489B E#3#4ED3 (##352)          MOVRM   R3,DA$RC                    ;
#489D F8#FFFCE (##353)          MOVLM   $7,SYS$FLGS                 ;RESET G3
#489F 4E36     (##354)          SUBRR   R3,R3
#48A# E#3##466 (##355)          MOVRM   R3,N$RC
#48A2 #E7#     (##356)          RETURN
#48A3 ####     (##357)          EVEN
               (##358) *
      ####48A4 (##359) CON1$ = #L
               (##36#) *
               (##361)          EJECT
```

```
(00362) *       ********************************************************
(00363) *       *                                                      *
(00364) *       *                  ADAPTIVE FILTERS AND                *
(00365) *       *       PITCH PERIOD AND CORRELATION CALCULATION       *
(00366) *       *                                                      *
(00367) *       ********************************************************
(00368) *
(00369) *       IT USES V3.5 MAP-EXECUTIVE
(00370) *
(00371) *       * DESCRIPTION:
(00372) *       THIS PROGRAM CALCULATES THE PITCH PERIOD AND CORRELATION COEFFICIENT
(00373) *       OF A BLOCK SAMPLES.  ALSO, DEPENDING ON THE CONDITION OF THE SAMPLE
(00374) *       BUFFER IN TRANSMITTER, IT WILL EMPLOY ADAPTIVE FILTER AND/OR PITCH
(00375) *       REPETITION.
(00376) *
(00377) *       * THIS PROGRAM EMPLOYS THE FOLLOWING BUFFERS:
(00378) *
(00379) *       1. ADAM BUFFER: ADAM SAMPLES INPUT ANALOG SIGNALS AND PUTS THE
(00380) *          CORRESPONDING QUANTIZED DIGITAL SIGNALS INTO THIS BUFFER.
(00381) *          --- DOUBLE BUFFERING --- SHORT FLOATING FORMAT ---
(00382) *       2. SAMPLE BUFFER: THE INPUT DATA COME FROM ADAM BUFFER. THE PURPOSE
(00383) *          OF THIS BUFFER IS TO PROVIDE BUFFER CONTROL TECHNIQUE.
(00384) *          --- CIRCULAR BUFFER --- SHORT FLOATING FORMAT ---
(00385) *       3. VHAT BUFFER: THIS BUFFER CONTAINS RECONSTRUCTED REDUCED SPEECH
(00386) *          SAMPLES FROM RARC-TRANSMITTER. ( THIS PROGRAM ONLY DEFINES THE
(00387) *          STARTING LOCATION OF VHAT POINTER.)
(00388) *
(00389) *       * SYMBOLIC DEFINITIONS:
(00390) *
(00391) *         TBLK:   TRANSFERING BLOCK SIZE FROM ADAM BUFFER TO SAMPLE BUFFER
(00392) *         FBLK:   FILTERING BLOCK SIZE
(00393) *         KBLK:   CORRELATION BLOCK SIZE
(00394) *         IEND:   UPPER LIMIT OF PITCH PERIOD
(00395) *         SSTX:   DATA POINTER IN SAMPLE BUFFER FOR USING IN PARC-TRANSMITTER
(00396) *         DASTX:  DATA POINTER IN SAMPLE BUFFER FOR THE TRANSFER FROM ADAM
(00397) *                 BUFFER
(00398) *         VHSTX:  VHAT POINTER IN VHAT BUFFER
(00399) *         INSGTH: LOCATION OF PITCH REPETITION THRESHOLD
(00400) *         INSFTH: LOCATION OF FILTER THRESHOLD
(00401) *         INSFBS: LOCATION OF FBLK-1
(00402) *         INSCST: LOCATION OF LOWER LIMIT OF PITCH PERIOD
(00403) *         INSCSZ: LOCATION OF CORRELATION BLOCK SIZE -1
(00404) *         INSGSZ: LOCATION OF PITCH REPETITION SIZE
(00405) *
```

```
                (00406) *       * RESTRICTIONS:
                (00407) *
                (00408) *       1. TBLK SHOULD BE MULTIPLE OF 2. THIS RESTRICTION CAN BE REMOVED
                (00409) *          IF THOSE STATEMENTS MARKED **A** ARE SLIGHTLY MODIFIED. THE PURPOSE
                (00410) *          OF THIS RESTRICTION IS TO REDUCE THE DATA TRANSFERING TIME.
                (00411) *       2. IT IS ASSUMED THAT DASTX NEVER OVERRIDES SSTX. THAT IS CONTROLLED BY
                (00412) *          BY PITCH REPETITION THRESHOLD AND SIZE.
                (00413) *       3. KBLK SHOULD BE MULTIPLE OF 4.
                (00414) *       4. THE ABSOLUTE VALUES OF UPPER LIMIT AND LOWER LIMIT OF CORRELATION
                (00415) *          COEFFICIENT SHOULD BE THE SAME.
                (00416) *       5. THE SEQUENCE OF INPUT SCALARS ARE AS FOLLOWS:
                (00417) *          TBLK/2,A,B,FBLF-0.5,KBLK-0.5,SEARCHING SIZE-0.5,2**-15,
                (00418) *          IEND-0.5,0.0001,ISTART,LB,UB,HL/(UB*2**15),HL/(UB*2**24),
                (00419) *          2**24*UB/HL
                (00420) *       6. THE LOCATIONS OF ARRAYS ARE AS FOLLOWS:
                (00421) *
                (00422) *          SAMPLE BUFFER: 0(3) ... 1023(3)
                (00423) *          VHAT BUFFER: 0(2) ... 1023(2) --- THE SIZE AND STARTING LOCATION
                (00424) *                       OF THIS BUFFER CAN BE CHANGED BY SLIGHTLY MODIFICATIONS
                (00425) *                       IN TRANSMITTER PROGRAMS.
                (00426) *
                (00427) *       * CALL THIS FUNCTION:
                (00428) *
                (00429) *         MAP = PICH3(SA,U)
                (00430) *         WHERE   SA --- SCALAR # FOR PITCH PERIOD
                (00431) *                 U  --- BUFFER # FOR ADAM BUFFER
                (00432) *
                (00433) *       CONTAINS FOR ASSEMBLY
                (00434) *
00000000        (00435)         M$S=0
00000794        (00436)         DMY$=$794                       ;DUMMY OUTPUT SPACE
00000001        (00437)         BUS1$=$0001                     ;BUS 1
00000288        (00438)         TOE$PRT=$288                    ;TOP OF EXEC.
000021FC        (00439)         CSPU$NOS=$21FC                  ;LOCATION OF NO-CSPU SUPPORT
00008E8         (00440)         AFDT$=$8E8                      ;STARTING ADDRESS OF AFDT
                (00441) *
00003E6         (00442)         SVT50$=SVT$+100                 ;SCALAR TABLE 50
00003E8         (00443)         SVT51$=SVT50$+2
00003EE         (00444)         SVT54$=SVT51$+6
00003F2         (00445)         SVT56$=SVT54$+4
00004B4         (00446)         SVT63$=SVT56$+18                ;ADD TWO SCALAR
00004BE         (00447)         SVT68$=SVT63$+10
00000412        (00448)         SVT70$=SVT68$+4
0000041A        (00449)         SVT74$=SVT70$+8
```

```
                00000426 (00450)          SVT79$=SVT74$+12                ;ADD ONE SCALAR
                0000042E (00451)          SVT83$=SVT79$+8
                00000446 (00452)          SVT98$=SVT83$+24
                0000044A (00453)          SVT99$=SVT98$+4                 ;ADD UPSAMPLING
                00000464 (00454)          SVT111$=SVT99$+26               ;ADD ONE SCALAR
                00000430 (00455)          NSTX=SVT83$+2
                00000466 (00456)          NSRC=SVT111$+2
                00000468 (00457)          SVT115$=NSRC+2
                0000046C (00458)          SVT117$=SVT115$+4
                0000046E (00459)          SVT118$=SVT117$+2
                         (00460) *        DISPATCH TABLE ENTRY
                         (00461) *
                000000F0 (00462)          FCB24=24                        ;FCB=24
                00000B88 (00463)          #L=AFDT$+3*2*(FCB24-128)        ;STARTING LOCATION OF AFDT FOR 24
                         (00464) *
          0B88 001E48A6 (00465)          ADDR      PICH3$(R7,1)          ;APU=PICH3
          0B8A 001E4A0E (00466)          ADDR      VPICH3$(R7,1)         ;APS=VPICH3
          0B8C 001021FC (00467)          ADDR      CSPU$NOS(,1,0)        ;NO CSPU
                         (00468) *
                         (00469) *        APU & APS MODULE
                         (00470) *
                0000048A4 (00471)          #L=CON1$
                         (00472) *
                         (00473) *        APU MODULE
                         (00474) *
                         (00475)          EVEN
          048A4 0000     (00476)          DATA      PICH3$SA              ;STARTING AADRESS
          048A5 0000     (00477)          DATA      PICH3$SZ              ;MODULE SIZE
                         (00478) *
                         (00479) PICH3$   BEGIN     APU(PICH3)            ;START OF APU MODULE
                00000003 (00480)          #M=3                            ;MAP-300
                00000000 (00481)          #A=0
                         (00482) *
                         (00483) *        DATA TRANSFER FROM ADAM BUFFER TO SAMPLE BUFFER
                         (00484) *
                         (00485) *        REGISTERS AFFECTED:
                         (00486) *
                         (00487) *        ADM1: A0,A1,A7,M0,M1,M7
                         (00488) *        ADM2: A1,A7,M0,M1,M7
                         (00489) *
                         (00490) *        INPUT SEQUENCE: TBLK/2, DATA FROM ADAM BUFFER
                         (00491) *        OUTPUT SEQUENCE: DATA TO SAMPLE BUFFER
                         (00492) *
     A00 048A6 1680168A (00493) PICH3$SA K(1)                           ;
```

```
PAGE   15:        MAP MODULES FOR THE  P A R C  ALGORITHM     --- JAN. 30, 1980

  A01 0048A0 00F00000 (00494)          MOV(IQA,A0) \ NOP              ; TBLK/2
  A02 0048AA 00E800E8 (00495)          MOV(IQA,M0)                    ;GAIN FACTOR*2**-13
  A03 0048AC 00E900E9 (00496)          MOV(IQA,M1)                    ;1/GAIN FACT*2**-13
  A04 0048AE 00970097 (00497)          MOV(R,A7)                      ; A7=1
  A05 0048B0 00EF0000 (00498) LOOP2    MOV(IQA,M7) \ NOP              ; THIS LOOP TO MASK
  A06 0049B2 000000EF (00499)          NOP \ MOV(IQA,M7)              ;      OFF LOWER BITS OFF
  A07 0048B4 0046E046 (00500)          MUL(M0,M7)                     ;           THE INP DATA
  A08 0048B6 00B100B1 (00501)          MOV(P,A1)                      ;
  A09 0048B8 3A203A20 (00502)          ALIGN(A1)                      ;
  A0A 0048BA 00910091 (00503)          MOV(R,A1)                        ;
  A0B 0048BC 32203220 (00504)          NORM(A1)                       ;
  A0C 0048BE 00F800F8 (00505)          MOV(R,M7)                      ;
  A0D 0048C0 04E804E8 (00506)          MUL(M1,M7)                     ;
  A0E 0048C2 4F000000 (00507)          SUB(A0,A7) \ NOP               ; DECREMENT COUNTER
  A0F 0048C4 00900000 (00508)          MOV(R,A0) \ NOP                ;
  A10 0048C6 00BC0000 (00509)          MOV(P,OQ) \ NOP
  A11 0048C8 000000BC (00510)          NOP \ MOV(P,OQ)
  A12 0048CA 901E0005 (00511)          JUMPC(LOOP2,T1)                ; LOOPING
                      (00512) *
                      (00513) *        ADAPTIVE FILTERS
                      (00514) *
                      (00515) *        REGISTERS AFFECTED:
                      (00516) *
                      (00517) *        ADM1: A0,A1,A2,M0,M6,M7,EXO
                      (00518) *        ADM2: A0,A1,M0,M6,EXO
                      (00519) *
                      (00520) *        FLAGS AFFECTED: AF1
                      (00521) *        FLAGS SENSED: G0,AF1
                      (00522) *
                      (00523) *        INPUT SEQUENCE: A,S(K-1),B,FBLK,S(K),SF(K-1),S(K),SF(K-1),....,DUMMY,
                      (00524) *                        DUMMY
                      (00525) *        OUTPUT SEQUENCE: SF(K),...
                      (00526) *
                      (00527) *        IT EMPLOYES IN PLACE OPERATION.
                      (00528) *
  A13 0048CC 00000000 (00529) AWAIT    NOP                            ;
  A14 0048CE 91040029 (00530)          JUMPS(PERIOD,G0)               ; NO FILTER IS EMPLOYED
  A15 0048D0 90090013 (00531)          JUMPC(AWAIT,AF1)               ; WAIT FOR SIGNAL FROM APS
  A16 0048D2 20202029 (00532)          CLEAR(AF1)                     ;
  A17 0048D4 00E80000 (00533)          MOV(IQA,M0) \ NOP              ; M0=A
  A18 0048D6 00EF0000 (00534)          MOV(IQA,M7) \ NOP              ; M7=S(K-1)
  A19 0048D8 0046E0E8 (00535)          MUL(M0,M7) \ MOV(IQA,M0)       ;                    M0=B
  A1A 0048DA 00F20000 (00536)          MOV(IQA,A2) \ NOP              ; A2=FILTERING BLOCK SIZE
  A1B 0048DC 00EE0000 (00537)          MOV(IQA,M6) \ NOP              ; M6=S(K)
```

```
A1C 048DE 00000EE (00538)          NOP \ MOV(IQA,M6)                ;                          M6=SF(K-1)
A1D 048E0 04588440 (00539)         MOV(EXO),MUL(M0,M6) \ MUL(M0,M6) ; EXO=A*S(K-1)
A1E 048E2 00000858 (00540)         NOP \ MOV(EXI,EXO)               ;
                   (00541) *
                   (00542) *       FILTERING LOOP
                   (00543) *
A1F 048E4 00B100B1 (00544) LOOP1   MOV(P,A1)                        ;
A20 048E6 00EE0000 (00545)         MOV(IQA,M6) \ NOP                ; M6=S(K)
A21 048E8 005000EE (00546)         MOV(EXI,A0) \ MOV(IQA,M6)        ;                          M6=SF(K-1)
A22 048EA 04588440 (00547)         MOV(EXO),MUL(M0,M6) \ MUL(M0,M6) ;
A23 048EC 41000858 (00548)         ADD(A0,A1) \ MOV(EXI,EXO)        ; A*S(K)+A*S(K-1)
A24 048EE 00980000 (00549)         MOV(R,EXO) \ NOP                 ;
A25 048F0 4F400850 (00550)         SUB(A2,A7) \ MOV(EXI,A0)         ;
A26 048F2 00004900 (00551)         NOP \ SUB(A0,A1)                 ;
A27 048F4 00920890 (00552)         MOV(R,A2) \ MOV(R,OQ)            ;
A28 048F6 9001E001F (00553)        JUMPC(LOOP1,T1)                  ;
                   (00554) *
                   (00555) *       INITIALIZATION FOR SEARCHING PITCH PERIOD
                   (00556) *
                   (00557) *       REGISTERS TO BE USED LATER:
                   (00558) *
                   (00559) *       ADM1: A3,A6,EXO
                   (00560) *       ADM2: A4,A6
                   (00561) *
                   (00562) *       INPUT SEQUENCE: KBLK/4,SEARCHING SIZE - 0.5
                   (00563) *
                   (00564) *       FLAGS AFFECTED: G0
                   (00565) *
A29 048F8 00D30000 (00566) PERIOD  MOV(IQ,A3) \ NOP                 ;A3=KBLK/4                 -----
A2A 048FA 00F80000 (00567)         MOV(IQA,EXO) \ NOP               ;EXO=KBLK/4
A2B 048FC 20242024 (00568)         CLEAR(G0)                        ;
A2C 048FE 000000F4 (00569)         NOP \ MOV(IQA,A4)                ;-----                     A4=SEARCHING SIZE F
A2D 04900 00160016 (00570)         MOV(ZERO,A6)                     ;      A6=SUM3
                   (00571) *
                   (00572) *       FIND T
                   (00573) *
                   (00574) *       REGISTERS AFFECTED:
                   (00575) *
                   (00576) *       ADM1: A0,A1,A2,A3,A5,A6
                   (00577) *       ADM2: A0,A1,A2,A4,A5,A6
                   (00578) *
                   (00579) *       INPUT SEQUENCE: S(J-T),S(J-T),S(J),S(J),(S(J-T),S(J-T),S(J),S(J))
                   (00580) *
                   (00581) *       FLAGS AFFECTED: AF0,AF1,AF3
```

```
                    (00582) *       FLAGS SENSED: AF1,AF3
                    (00583) *
  A2E 0490 2040490 (00584)          SET(AF1)                          ;FIRST SEARCHING
  A2F 0494 2048240 (00585)          SET(AF0)                          ;SIGNAL APS TO PRODUCE DATA
  A30 0496 80F10000 (00586) LOOP0   MOV(IQA,A1) \ NOP                 ;A1=S(J-T)
  A31 0498 000008F1 (00587)         NOP \ MOV(IQA,A1)
  A32 049A 80F00000 (00588)         MOV(IQA,A0) \ NOP                 ;A0=S(J)
  A33 049C 000008F0 (00589)         NOP \ MOV(IQA,A0)
                    (00590) *
  A34 049E 49004900 (00591) LOOP6   SUB(A0,A1)                        ;     R=S(J)-S(J-T)
  A35 0490 12B512B5 (00592)         MOV(A5),ABS(A5)                   ;     R=ABS(S(J)-S(J-T))
  A36 0492 80F10000 (00593)         MOV(IQA,A1) \ NOP                 ;
  A37 0494 000008F1 (00594)         NOP \ MOV(IQA,A1)                 ;
  A38 0496 46B546B5 (00595)         MOV(A5),ADD(A5,A6)                ;     A6=SUM3
  A39 0498 80F00000 (00596)         MOV(IQA,A0) \ NOP
  A3A 049A 000008F0 (00597)         NOP \ MOV(IQA,A0)                 ;
                    (00598) *
  A3B 049C 49164916 (00599)         MOV(A6),SUB(A0,A1)                ;      R=S(J)-S(J-T)
  A3C 049E 12B512B5 (00600)         MOV(A5),ABS(A5)                   ;
  A3D 0500 80F10000 (00601)         MOV(IQA,A1) \ NOP                 ;
  A3E 0502 000008F1 (00602)         NOP \ MOV(IQA,A1)                 ;
  A3F 0504 46B546B5 (00603)         MOV(A5),ADD(A5,A6)                ;
  A40 0506 80F00000 (00604)         MOV(IQA,A0) \ NOP                 ;
  A41 0508 000008F0 (00605)         NOP \ MOV(IQA,A0)                 ;
  A42 050A 4F760896 (00606)         MOV(A6),SUB(A3,A7) \ MOV(R,A6)    ;R=KBLK/4 - 1
  A43 050C 80930898 (00607)         MOV(R,A3) \ MOV(R,EXO)            ;
  A44 050E 9010E004 (00608)         JUMPC(LOOP6,T1)                   ;
                    (00609) *
  A45 0510 00554F00 (00610)         MOV(EXI,A5) \ SUB(A4,A7)          ;-----                  R=SEARCHING SIZE-1
  A46 0512 46A00858 (00611)         ADD(A5,A6) \ MOV(EXI,EXO)         ;R=SUM3                EXO=KBLK/4
  A47 0514 900904E (00612)          JUMPC(LOOP7,AF1)
                    (00613) *
  A48 0516 20202029 (00614)         CLEAR(AF1)
  A49 0518 00530894 (00615)         MOV(EXI,A3) \ MOV(R,A4)           ;A3=KBLK/4
  A4A 051A 00920892 (00616)         MOV(R,A2) \ MOV(R,A2)             ;A2=SMALL               A2=IEND-0.5-T
  A4B 051C 00150815 (00617)         MOV(ZERO,A5)
  A4C 051E 00160816 (00618)         MOV(ZERO,A6)
  A4D 0520 10000F2F (00619)         JUMP(LOOP0-1)
                    (00620) *
  A4E 0522 4E560000 (00621) LOOP7   MOV(A6),SUB(A2,A6) \ NOP          ;R=SMALL-SUM3
  A4F 0524 00530000 (00622)         MOV(EXI,A3) \ NOP                 ;A3=KBLK/4
  A50 0526 00150815 (00623)         MOV(ZERO,A5)
  A51 0528 00000894 (00624)         MOV(R,NULL) \ MOV(R,A4)           ;----                   A4=SEARCHING SIZE-1
  A52 052A 911E0058 (00625)         JUMPS(LOOP8,T1)
```

```
                    (00626) *
A53 0494C 2040204B (00627)          SET(AF3)                        ;SIGNAL APS TO RENEW T
A54 0494E 02C00892 (00628)          R(A6) \ MOV(R,A2)               ;
A55 04950 0892C000 (00629)          MOV(R,A2) \ NOP                 ;A2=SMALL
A56 04952 00000000 (00630) WAIT     NOP
A57 04954 910B0056 (00631)          JUMPS(WAIT,AF3)
                    (00632) *
A58 04956 00160016 (00633) LOOP8    MOV(ZERO,A6)
A59 04958 901F002F (00634)          JUMPC(LOOP0-1,T2)
                    (00635) *
                    (00636) *       CALCULATE CORRELATION COEFFICIENT
                    (00637) *
                    (00638) *       REGISTERS AFFECTED:
                    (00639) *
                    (00640) *       ADM1: A0,A3,A4,A5,A6,M0,M1,M2,M6,M7
                    (00641) *       ADM2: A0,A4,A5,A6,M0,M1,M2,M6,M7
                    (00642) *
                    (00643) *       INPUT SEQUENCE: S(J-T),S(J-T),(S(J),S(J),S(J-T),S(J-T),S(J),S(J),
                    (00644) *                       S(J-T),S(J-T))
                    (00645) *
                    (00646) *       FLAGS AFFECTED: AF2
                    (00647) *
A5A 0495A 2040204A (00648)          SET(AF2)                        ;SIGNAL APS TO STOP SEARCHING
A5B 0495C 00140014 (00649)          MOV(ZERO,A4)                    ;A4=SUM OF S(J-T)**2
A5C 0495E 00CF0000 (00650)          MOV(IQ,M7) \ NOP                ;M7=S(J-T)
A5D 04960 00E90000 (00651)          MOV(IQA,M1) \ NOP               ;M1=S(J-T)
A5E 04962 000000CF (00652)          NOP \ MOV(IQ,M7)                ;
A5F 04964 000000E9 (00653)          NOP \ MOV(IQA,M1)               ;
                    (00654) *
A60 04966 04E004E0 (00655) LOOP9    MUL(M1,M7)                      ;     P=S(J-T)**2
A61 04968 46A046A0 (00656)          ADD(A5,A6)                      ;     R=SUM1
A62 0496A 00E00000 (00657)          MOV(IQA,M0) \ NOP               ;M0=S(J)
A63 0496C 000000E0 (00658)          NOP \ MOV(IQA,M0)               ;
A64 0496E 047004F0 (00659)          MOV(A0),MUL(M0,M7)              ;     P=S(J)*S(J-T)
A65 04970 40960496 (00660)          MOV(A6),ADD(A4,A0)              ;     R=SUM2
A66 04972 00CE0000 (00661)          MOV(IQ,M6) \ NOP
A67 04974 00EA0000 (00662)          MOV(IQA,M2) \ NOP
A68 04976 000000CE (00663)          NOP \ MOV(IQ,M6)
A69 04978 000000EA (00664)          NOP \ MOV(IQA,M2)
A6A 0497A 85558555 (00665)          MOV(A5),MUL(M2,M6)
A6B 0497C 45B445B4 (00666)          MOV(A4),ADD(A5,A6)
A6C 0497E 00E00000 (00667)          MOV(IQA,M0) \ NOP
A6D 04980 000000E0 (00668)          NOP \ MOV(IQA,M0)
A6E 04982 045004F0 (00669)          MOV(A0),MUL(M0,M6)
```

```
A6F 04984 4009640096 (00670)        MOV(A6),ADD(A4,A0)
A70 04986 008CF0000 (00671)         MOV(IQ,M7) \ NOP
A71 04988 008E90000 (00672)         MOV(IQA,M1) \ NOP
A72 0498A 4F740894 (00673)          MOV(A4),SUB(A3,A7) \ MOV(R,A4)
A73 0498C 000000CF (00674)          NOP \ MOV(IQ,M7)
A74 0498E 000008E9 (00675)          NOP \ MOV(IQA,M1)
A75 04990 008B5008B5 (00676)        MOV(P,A5)
A76 04992 00930098 (00677)          MOV(R,A3) \ MOV(R,EXO)
A77 04994 901E0006 (00678)          JUMPC(LOOP9,T1)
                    (00679) *
                    (00680) *       OUTPUT BETA AND T
                    (00681) *
                    (00682) *       REGISTERS TO BE USED:
                    (00683) *
                    (00684) *       ADM1: A0,A1,A4,A7,M0,M2,M6,M7,EXO
                    (00685) *       ADM2: A0,A2,A4,A6,A7,M2,M6,EXO
                    (00686) *
                    (00687) *       INPUT SEQUENCE: 1/2**15,IEND-0.5,0.0001,ISTART,LOWER BETA LIMIT,
                    (00688) *                       UPPER BETA LIMIT
                    (00689) *       OUTPUT SEQUENCE: T(IN INTEGER)
                    (00690) *
A78 04996 0050008EA (00691)         MOV(EXI,A0) \ MOV(IQA,M2)       ;A0=SUM2            M2=1./(2.**15)
A79 04998 44000F4 (00692)           ADD(A0,A4) \ MOV(IQA,A4)        ;R=SUM2             A4=IEND-0.5
A7A 0499A 00F14A00 (00693)          MOV(IQA,A1) \ SUB(A4,A2)        ;A1=0.0001
A7B 0499C 000008F7 (00694)          NOP \ MOV(IQA,A7)               ;                   A7=ISTART
A7C 0499E 00004F94 (00695)          NOP \ MOV(A4),SUB(A4,A7)        ;                   ENCODE T
A7D 049A0 0000008E (00696)          NOP \ MOV(R,M6)                 ;                   M6=T
A7E 049A2 00008540 (00697)          NOP \ MUL(M2,M6)                ;
A7F 049A4 008F008F7 (00698)         MOV(R,M7) \ MOV(IQA,A7)         ;M7=SUM2            A7=LOWER BETA
A80 049A6 4C340084 (00699)          MOV(A4),SUB(A1,A4) \ MOV(P,A4) ;R=0.0001-SUM2
A81 049A8 008F73A8 (00700)          MOV(IQA,A7) \ ALIGN(A4)         ;A7=UPPER BETA
A82 049AA 0000089C (00701)          MOV(R,NULL) \ MOV(R,OQ)
A83 049AC 901E00A9 (00702)          JUMPC(RESET,T1)
                    (00703) *
A84 049AE 2E000000 (00704)          RCP(A4) \ NOP
A85 049B0 46A046A0 (00705)          MOV(M0),ADD(A5,A6) \ ADD(A5,A6) ;M0=F0              R=SUM1
A86 049B2 84600000 (00706)          MUL(M0,M7) \ NOP                ;P=F0*SUM2
A87 049B4 16800000 (00707)          MOV(EXO),K(2) \ NOP
A88 049B6 008000850 (00708)         MOV(P,A0) \ MOV(EXI,A0)         ;A0=F0*SUM2         A0=SUM1
A89 049B8 48314616 (00709)          MOV(A1),SUB(A1,A0) \ MOV(A6),ADD(A0,A6) ;R=2-A0     R=SUM1
A8A 049BA 008E008A (00710)          MOV(R,M6) \ MOV(R,M2)           ;M6=2-F0*SUM2       M2=SUM1
A8B 049BC 84400098 (00711)          MUL(M0,M6) \ MOV(R,EXO)         ;P=SUM2*M6          EXO=SUM1
A8C 049BE 004A0000 (00712)          MOV(EXI,M2) \ NOP               ;M2=SUM1
A8D 049C0 00B00000 (00713)          MOV(P,EXO) \ NOP                ;
```

```
A8E 049C2 08AE084E (00714)          MOV(P,M6) \ MOV(EXI,M6)          ;      M6=1/SUM2
A8F 049C4 0540854A (00715)          MUL(M2,M6)                       ;      P=SUM1/SUM2
A90 049C6 08800080 (00716)          MOV(P,A0)                        ;      A0=SUM1/SUM2
                  (00717) *
                  (00718) *         LIMIT OF BETA
                  (00719) *
                  (00720) *         INPUT SEQUENCE: HL*2**-15/UB,HL*2**-24/UB,UB*2**24/HL
                  (00721) *         OUTPUT SEQUENCE: QUANTIZED BETA(IN INTEGER),BETA
                  (00722) *         WHERE  HL --- HALF OF # OF QUANTIZER LEVEL FOR BETA
                  (00723) *                BETA --- CORRELATION COEFFICIENT OF A BLOCK OF SPEECH
                  (00724) *                UB --- UPPER LIMIT OF BETA
                  (00725) *
A91 049C8 68E06E0E (00726)          MIN(A7,A0) \ MAX(A7,A0)          ;LIMIT OF BETA
A92 049CA 08000080 (00727)          MOV(R,NULL) \ MOV(R,NULL)
A93 049CC 901F00AC (00728)          JUMPC(LOOP4,T2)
A94 049CE 08900000 (00729)          MOV(R,A0) \ NOP                  ;OUTPUT BETA
A95 049D0 47000000 (00730) LOOP5    ADD(A0,A7) \ NOP                 ;R=BETA+UB
A96 049D2 08EE0000 (00731)          MOV(IQA,M6)\ NOP                 ;M6=HL*2**(-15)/UB
A97 049D4 00000EE (00732)           NOP \ MOV(IQA,M6)                ;                    M6=HL*2**-24/UB
A98 049D6 08900000 (00733)          MOV(R,EXO) \ NOP                 ;
A99 049D8 080A084A (00734)          MOV(R,M2)  \ MOV(EXI,M2)         ;M2=BETA+UB
A9A 049DA 0540854A (00735)          MUL(M2,M6)                       ;READY FOR ALIGNMENT
A9B 049DC 08800080 (00736)          MOV(P,A0)                        ;A0=P
A9C 049DE 3A003A00 (00737)          ALIGN(A0)                        ;R=BETA ENCD.
A9D 049E0 00000EE (00738)           NOP \ MOV(IQA,M6)                ;M6=2**24*UB/HL
A9E 049E2 08900210 (00739)          MOV(R,OQ)  \ MOV(A0),NORM(A0)    ;OUT FIXED ENCD. BETA
A9F 049E4 0000080A (00740)          NOP \ MOV(R,M2)                  ;M2=NORM BETA
AA0 049E6 00008540 (00741)          NOP \ MUL(M2,M6)                 ;P=(BETA+UB) DIGITIZED
AA1 049E8 00000080 (00742)          NOP \ MOV(P,EXO)
AA2 049EA 08500000 (00743)          MOV(EXI,A0) \ NOP
AA3 049EC 4F000000 (00744)          SUB(A0,A7) \ NOP                 ;R=BETA DIGITIZED
AA4 049EE 08900000 (00745)          MOV(R,OQ)  \ NOP                 ;OUT BETA
AA5 049F0 20322032 (00746)          CLEAR(RA)
AA6 049F2 00000000 (00747)          NOP
AA7 049F4 10000000 (00748)          JUMP(0)                          ;JUMP TO BEGINNING
AA8 049F6 00000000 (00749)          NOP
AA9 049F8 00100000 (00750) RESET    MOV(ZERO,A0) \ NOP
AAA 049FA 10000095 (00751)          JUMP(LOOP5)
AAB 049FC 00000000 (00752)          NOP
                  (00753) *
AAC 049FE 00000098 (00754) LOOP4    NOP \ MOV(R,EXO)
AAD 04A00 00500000 (00755)          MOV(EXI,A0) \ NOP
AAE 04A02 10000095 (00756)          JUMP(LOOP5)
AAF 04A04 00000000 (00757)          NOP
```

```
      00000B0B (00758) PICH3$SZ=.A-PICH3$SA                ;COMPUTE THE SIZE OF THE MODULE
0A06            (00759)         END
                (00760) *
                (00761)         EJECT
```

```
                    (00762) *
                    (00763) *       APS MODULE OF PITCH EXTRACTION PROGRAM ----- VPICH3
                    (00764) *
                    (00765)         EVEN                            ;START ON WORD BOUNDARY
     04A06 0004AF6  (00766)         ADDR    VPICH3$I                ;PTR TO CONSTR INSTR BLOCK
     04A08 0004ACE  (00767)         ADDR    VPICH3$+2*VPICH3$S      ;STARTING ADDRESS OF SCALAR
     04A0A 0001     (00768)         DATA    1                       ;ONE SCALAR
     04A0B 00EA     (00769)         DATA    VPICH3$Z                ;MODULE SIZE
     04A0C 0004A16  (00770)         ADDR    VPICH3$A                ;CHAIN ANCHOR
                    (00771)         EVEN
                    (00772) *
                    (00773) VPICH3$ BEGIN   APS(VPICH3)
                    (00774) *
                    (00775) *       INITIATE APU
                    (00776) *
 A00 04A0E 03000032 (00777)         SET(RA)                         ; RUN APU
                    (00778) *
                    (00779) *       DATA TRANSFER FROM ADAM BUFFER TO SAMPLE BUFFER
                    (00780) *
                    (00781) *       REGISTERS AFFECTED: BR0,BR1,BR2,BR3,BW1,BW2
                    (00782) *
 A01 04A10 02F203E6 (00783)         LOAD(BR3,SVT50$(1),L,TF)        ; TBLK/2
 A02 04A12 04F2046C (00784)         LOAD(BR3,SVT117$(1),L,TF)       ; GAIN FACTOR
 A03 04A14 06BA0004 (00785)         ADD(BR3,4,TF)                   ; INV GAIN FACT.
 A04 04A16 00701000 (00786)         LOAD(BR3,[1])                   ; ADDRESS POINTER FOR ADAM BUFFER - 1
 A05 04A18 0A600000 (00787)         LOAD(BR2,M$$)                   ; SIZE - 1
 A06 04A1A 0C320000 (00788)         SUB(BR3,M$$)
           0004A1D  (00789)         SSTX=@L+1                       ;MODIFY S POINTER
 A07 04A1C 0E4E0000 (00790)         LOAD(BR0,@(3),S)                ;
 A08 04A1E 10110011 (00791)         MOVB(BW1,BR0)                   ;S$TX (+N+G) S(K-1)
           0004A21  (00792)         DA$TX=@L+1                      ;MODIFY DATA POINTER
 A09 04A20 124E0001 (00793)         LOAD(BR0,1(3),S)                ;
 A0A 04A22 14210011 (00794)         MOVB(BW2,BR0)                   ;SD$TX (+S) SD(K-1)
           0004A25  (00795)         VH$TX=@L+1                      ;MODIFY VHAT POINTER
 A0B 04A24 165C0000 (00796)         LOAD(BR1,@(2),S)                ;VHAT(K) (+N) VH$TX
 A0C 04A26 184003FF (00797)         LOAD(BR0,1023)                  ; MASK FOR SAMPLE BUFFER **
 A0D 04A28 1A291A79 (00798)         ADDL(BR2,1),JUMP(VLOOP3)        ;
 A0E 04A2A 1CBA0001 (00799) VLOOP2  ADD(BR3,1,TF)                   ;
 A0F 04A2C 1E210039 (00800)         ADDL(BW2,1)                     ;
 A10 04A2E 20A10001 (00801)         ANDB(BW2,BR0,TF)                ; OUTPUT ADDRESS IN SAMPLE BUFFER
 A11 04A30 22BA0001 (00802)         ADD(BR3,1,TF)                   ;
 A12 04A32 24210039 (00803)         ADDL(BW2,1)                     ;
 A13 04A34 26A10001 (00804)         ANDB(BW2,BR0,TF)                ; OUTPUT ADDRESS IN SAMPLE BUFFER
 A14 04A36 28BA0001 (00805)         ADD(BR3,1,TF)                   ;
```

```
A15 04A38 2A210039 (00086)          ADDL(BW2,1)                    ;
A16 04A3A 2CA10001 (00087)          ANDB(BW2,BR0,TF)               ; OUTPUT ADDRESS IN SAMPLE BUFFER
A17 04A3C 2E8A0001 (00088)          ADD(BR3,1,TF)                  ;
A18 04A3E 30210039 (00089)          ADDL(BW2,1)                    ;
A19 04A40 32A10001 (00810)          ANDB(BW2,BR0,TF)               ; OUTPUT ADDRESS IN SAMPLE BUFFER
A1A 04A42 3429EB4 (00811) VLOOP3    SUBL(BR2,4),JUMPP(VLOOP2)      ;
A1B 04A44 36292FF8 (00812)          ADDL(BR2,3),JUMPN(CKP)         ;
A1C 04A46 388A0001 (00813) VLOOP4   ADD(BR3,1,TF)                  ;
A1D 04A48 3A210039 (00814)          ADDL(BW2,1)                    ;
A1E 04A4A 3CA10001 (00815)          ANDB(BW2,BR0,TF)               ; OUTPUT ADDRESS IN SAMPLE BUFFER
A1F 04A4C 3E291CB1 (00816)          SUBL(BR2,1),JUMPP(VLOOP4)      ;
                   (00817) *
                   (00818) *        CHECK ADAPTIVITY (INCLUDING FILTERING AND PITCH REPETITION)
                   (00819) *
                   (00820) *        REGISTERS PERMENENTLY USED BY THIS PROGRAM:
                   (00821) *
                   (00822) *        BW2=THE POINTER FOR ADAM OUTPUT ADDRESS
                   (00823) *        BW1=THE POINTER FOR PARC INPUT ADDRESS
                   (00824) *
                   (00825) *        REGISTERS AFFECTED: BR0,BR2,BR3
                   (00826) *
                   (00827) *        FLAGS AFFECTED: G2,AF1
                   (00828) *
         0004A4F (00829)            INSGTH=0L+1                    ;INITIALIZE PITCH REPETITION THRESHOLD
A20 04A4E 4070028A (00830) CKP      LOAD(BR3,650)                  ; PITCH REPETITION THRESHOLD
A21 04A50  2290014 (00831)          MOVB(BR2,BW2)                  ;
A22 04A5. 042924A2 (00832)          SUBB(BR2,BW1),JUMPP(CONT1)     ;
A23 04A54 462A0400 (00833)          ADD(BR2,1024)                  ;**
         0004A57 (00834)            INSFTH=0L+1                    ;INITIALIZE FILTER THRESHOLD
A24 04A56 4040001F4 (00835) CONT1   LOAD(BR0,500)                  ; NO FILTER THRESHOLD
A25 04A58 4A2927E7 (00836)          SUBB(BR2,BR3),JUMPN(NEXT1)     ; IF < 0 => PITCH REPETITION
A26 04A5A 4C300026 (00837)          SET(G2)                        ; => PITCH REPETITION
A27 04A5C 4E29002F (00838) NEXT1    ADDB(BR2,BR3)                  ;
A28 04A5E 5029DE1 (00839)           SUBB(BR2,BR0),JUMPN(NOFTR)     ; IF > 0 => NO FILTER IS EMPLOYED
A29 04A60 52300029 (00840)          SET(AF1)                       ;SIGNAL APU --- FILTERING
                   (00841) *
                   (00842) *
                   (00843) *        FILTER
                   (00844) *
                   (00845) *        REGISTERS AFFECTED: BR0,BR2,BR3,BW0
                   (00846) *
A2A 04A62 54C203E8 (00847) FILTER   LOAD(BR0,SVT51$(1),L,TF)        ;A ADDRESS
A2B 04A64 56A90012 (00848)          MOVB(BR2,BW1,TF)               ;S(K-1)
A2C 04A66 588A0002 (00849)          ADD(BR0,2,TF)                  ;B
```

```
A2D 04A60 5A0A0002 (00850)          ADD(BR0,2,TF)                    ;FILTERING BLOCK
          0000A6B (00851)           INSFBS=$L+1                      ;INI. FILT. BLK. SIZE
A2E 04A6A 5C400A4F (00852)          LOAD(BR0,79)                     ;FILTERING SIZE
A2F 04A6C 5E700FF (00853)           LOAD(BR3,1023)                   ; MASK**
A30 04A6E 60010015 (00854)          MOVB(BW0,BR2)                    ;SF(K-1) POINTER
A31 04A70 622A0001 (00855)          ADD(BR2,1)
A32 04A72 64A90007 (00856)          ANDB(BR2,BR3,TF)                 ; S(K)
A33 04A74 66220001 (00857)          SUB(BR2,1)                       ;
A34 04A76 68A90007 (00858)          ANDB(BR2,BR3,TF)                 ; SF(K-1)
                   (00859) *
A35 04A78 6A2A0002 (00860) VLOOP1   ADD(BR2,2)                       ;
A36 04A7A 6CA90007 (00861)          ANDB(BR2,BR3,TF)                 ; S(K)
A37 04A7C 6E220001 (00862)          SUB(BR2,1)                       ;
A38 04A7E 70A90007 (00863)          ANDB(BR2,BR3,TF)                 ; SF(K-1)
A39 04A80 72010039 (00864)          ADDL(BW0,1)                      ;
A3A 04A82 74810007 (00865)          ANDB(BW0,BR3,TF)                 ; SF(K)
A3B 04A84 76093581 (00866)          SUBL(BR0,1),JUMPP(VLOOP1)        ; LOOPING
A3C 04A86 78003E60 (00867)          JUMP(NOFTR+1)
                   (00868) *
                   (00869) *        PITCH PERIOD AND CORRELATION COEFFICIENT CALCULATIONS
                   (00870) *
                   (00871) *        REGISTERS AFFECTED: BR0,BW0,BW3
                   (00872) *
                   (00873) *        FLAGS AFFECTED: G0
                   (00874) *
A3D 04A88 7A300024 (00875) NOFTR    SET(G0)                          ; SIGNAL APU --- NO FILTERING
A3E 04A8A 7CC203EE (00876)          LOAD(BR0,SVT54$(1),L,TF)         ;KBLK/4
A3F 04A8C 7E8A0002 (00877)          ADD(BR0,2,TF)                    ;CORRELATION SIZE
          0000A8F (00878)           INSCST=$L+1                      ;INI. LOWER LIMIT OF T
A40 04A8E 80400013 (00879)          LOAD(BR0,19)                     ;STARTING # OF CORRELATION
A41 04A90 82010011 (00880)          MOVB(BW0,BR0)                    ;INI VALUE OF T
A42 04A92 84310011 (00881)          MOVB(BW3,BR0)                    ;
                   (00882) *
                   (00883) *        SEARCHING T
                   (00884) *
                   (00885) *        REGISTERS AFFECTED: BR0,BR2,BR3,BW0
                   (00886) *
                   (00887) *        FLAGS AFFECTED: AF0,AF3
                   (00888) *        FLAGS SENSED: AF0,AF2,AF3
                   (00889) *
A43 04A94 86000FEA (00890) IN4     JUMPS(FINISH,AF2)                ;JUMP IF FINISH PITCH PERIOD SEARCHING
A44 04A96 88000BEB (00891)         JUMPS(RECORD,AF3)                ;RECORD T
A45 04A98 8A004A8 (00892)          JUMPC(IN4,AF0)                   ;WAIT FOR SIGNAL
A46 04A9A 8C200028 (00893)         CLEAR(AF0)
```

```
PAGE   25:        MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980

  A47 04A9C 0E4003FF (00894) IN2    LOAD(BR0,1023)                  ; MASK OF SAMPLE BUFFER**
      00004A9F (00895)              IN$CSZ=@L+1                     ;INI. CORR. BLK. SIZE
  A48 04A9E 9070004F (00896)        LOAD(BR3,79)                    ; CORRELATION BLOCK SIZE - 1
            (00897) ;                                   ASSUME TRANSFER BLOCK >= CORRELATION BLOCK SIZE
  A49 04AA0 92290012 (00898)        MOVB(BR2,BW1)                   ; STARTING ADDRESS
            (00899) *
  A4A 04AA2 94290026 (00900) IN1     SUBB(BR2,BW3)                  ;S(J-IT)
  A4B 04AA4 96A90001 (00901)         ANDB(BR2,BR0,TF)
  A4C 04AA6 982A0001 (00902)         ADD(BR2,1)
  A4D 04AA8 9AA90001 (00903)         ANDB(BR2,BR0,TF)
  A4E 04AAA 9C29002E (00904)         ADDB(BR2,BW3)
  A4F 04AAC 9EA90001 (00905)         ANDB(BR2,BR0,TF)
  A50 04AAE A02A0001 (00906)         ADD(BR2,1)
  A51 04AB0 A2A90001 (00907)         ANDB(BR2,BR0,TF)
  A52 04AB2 A4394AB2 (00908)         SUBL(BR3,2),JUMPP(IN1)         ;JUMP IF NOT END
            (00909) *
  A53 04AB4 A6C20794 (00910)         LOAD(BR0,DMY$(1),L,TF)
  A54 04AB6 A8C20794 (00911)         LOAD(BR0,DMY$(1),L,TF)
  A55 04AB8 AA005AEA (00912)         JUMPS(CLOSE,AF2)
  A56 04ABA ACC20794 (00913)         LOAD(BR0,DMY$(1),L,TF)
  A57 04ABC AEC20794 (00914)         LOAD(BR0,DMY$(1),L,TF)
  A58 04ABE B0310039 (00915)         ADDL(BW3,1)
  A59 04AC0 B2004360 (00916)         JUMP(IN4)
            (00917) *
            (00918) *        OUTPUT BETA & T
            (00919) *
            (00920) *        REGISTERS AFFECTED: BR0,BR3,BW3
            (00921) *
            (00922) *        FLAGS AFFECTED: AF2
            (00923) *
  A5A 04AC2 B4C203F2 (00924) CLOSE   LOAD(BR0,SVT56$(1),L,TF)       ;1./(2.**15)
  A5B 04AC4 B68A0002 (00925)         ADD(BR0,2,TF)                  ;IEND-0.5
  A5C 04AC6 B88A0002 (00926)         ADD(BR0,2,TF)                  ;0.0001
  A5D 04AC8 BA8A0002 (00927)         ADD(BR0,2,TF)                  ;ISTART
  A5E 04ACA BC8A0002 (00928)         ADD(BR0,2,TF)                  ;LOWER BETA LIMIT
  A5F 04ACC BE8A0002 (00929)         ADD(BR0,2,TF)                  ;UPPER BETA LIMIT
  A60 04ACE C0720000 (00930) VPICH3$S LOAD(BR3,MS$(1),L)            ;
  A61 04AD0 C3310017 (00931)         MOVB(BW3,BR3,TE)               ;OUTPUT T (IN INTEGER)
  A62 04AD2 C48A0002 (00932)         ADD(BR0,2,TF)                  ;HL*2**(-15)/UB
  A63 04AD4 C68A0002 (00933)         ADD(BR0,2,TF)                  ;HL*2**-24/UB
  A64 04AD6 C88A0002 (00934)         ADD(BR0,2,TF)                  ;UB*2**24/HL
  A65 04AD8 CB31003A (00935)         ADDL(BW3,2,TE)                 ;OUTPUT INTEGER ENCD. BETA
  A66 04ADA CCB1003A (00936)         ADDL(BW3,2,TF)                 ;BETA
  A67 04ADC CE010039 (00937)         ADDL(BW0,1)
```

```
A68 04ADE D02000A (00938)          CLEAR(AF2)
A69 04AE0 D220031 (00939)          CLEAR(RI)
A6A 04AE2 D400002 (00940)          NOP(0)
                  (00941) *
A6B 04AE4 D620028 (00942) RECORD   CLEAR(AF3)
A6C 04AE6 D801016 (00943)          MOVB(BW0,BW3)
A6D 04AE8 DA01031 (00944)          SUBL(BW0,1)
A6E 04AEA DC00360 (00945)          JUMP(IN4)
                  (00946) *
                  (00947) *        CALCULATE BETA
                  (00948) *
                  (00949) *        REGISTERS AFFECTED: BR2,BW1,BW3
                  (00950) *
                  (00951) *        FLAGS SENSED: G2
                  (00952) *
A6F 04AEC DE3100F (00953) FINISH   MOVB(BW3,BW0)
A70 04AEE E000476 (00954)          JUMPC(IN2,G2)                  ; NO PITCH REPETITION
          0004AF1 (00955)          INSGSZ=*L+1                    ;INI. PITCH REPETITION SIZE
A71 04AF0 E260030 (00956)          LOAD(BR2,60)                   ; LOAD PITCH REPETITION SIZE
A72 04AF2 E411002 (00957)          ADDB(BW1,BR2)                  ; UPDATE S$TX
A73 04AF4 E600476 (00958)          JUMP(IN2)
                  (00959) *
          0004A16 (00960) VPICH3$A=*C                            ;CHAIN ANCHOR
    04AF6         (00961)          END
                  (00962) *
                  (00963)          EVEN
                  (00964) *
                  (00965) *        STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                  (00966) *
    04AF6 00000000 (00967) VPICH3$I    DATA    1F'0.0'
                  (00968) *
          000005EA (00969) VPICH3$Z=*L-VPICH3$                    ;COMPUTE MODULE SIZE
                  (00970) *
          0004AF8 (00971)          CON2$=*L
                  (00972) *
                  (00973)          EJECT
```

```
(00974) *       ****************************************************
(00975) *       *                                                  *
(00976) *       *      PITCH EXTRACTION ADAPTIVE RESIDUAL CODER    *
(00977) *       *                   P A R C                        *
(00978) *       *                                                  *
(00979) *       ****************************************************
(00980) *
(00981) *        IT USES V3.5 MAP-EXECUTIVE
(00982) *
(00983) *       *  DESCRIPTION:
(00984) *
(00985) *       THIS IS A SPEECH CODING PROGRAM IN MAP-300 ARITHMETIC PROCESSOR
(00986) *       FOR DIGITAL TRANSMISION OF SPEECH AT A RATE OF 9600 BITS PER SECOND.
(00987) *       THE ALGORITHM COMBINES PITCH EXTRACTION LOOP, PITCH COMPENSATING
(00988) *       ADAPTIVE QUANTIZER, SEQUENTIALLY ADAPTIVE PREDICTOR.
(00989) *
(00990) *       1. THE REDUNDANCY OF SPEECH IS REMOVED IN THE PITCH EXTRACTION LOOP
(00991) *          BLOCK BY BLOCK.
(00992) *       2. THE SEQUANTIALLY ADAPTIVE PREDICTOR USING BACKWARD ADAPTION
(00993) *          IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
(00994) *       3. THE ERROR IN THIS ESTIMATE IS NORMALIZED AND QUANTIZED BY THE
(00995) *          PITCH COMPENSATING ADAPTIVE QUANTIZER.
(00996) *       4. IF PITCH REPETITION MODE HAPPENS, A BLOCH OF SHAT IS DUPLICATED.
(00997) *
(00998) *       * SYMBOLIC ABBREVIATIONS:
(00999) *
(01000) *         INSTX    LOCATION OF PITCH REPETITION SIZE IN APS MODULE
(01001) *
(01002) *       * BUFFERS:
(01003) *
(01004) *       1. SAMPLE BUFFER: IT CONTAINS INPUT SPEECH SAMPLES AND FILTERED
(01005) *            INPUT SPEECH SAMPLES.
(01006) *          --- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
(01007) *       2. VHAT BUFFER: IT CONTAINS RECONSTRUCTED REDUCED SPEECH SAMPLES.
(01008) *          --- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
(01009) *       3. SHAT BUFFER: IT CONTAINS RECONSTRUCTED SPEECH SAMPLES.
(01010) *          --- CIRCULAR BUFFER --- SHORT REAL FORMAT ---
(01011) *       4. Q BUFFER: IT CONTAINS OUTPUT QUANTIZING LEVELS.
(01012) *          --- DOUBLE BUFFERS --- INTEGER * 2 FORMAT ---
(01013) *       5. PARAMETER BUFFER: IT CONTAINS PARAMETERS FOR QUANTIZER INCLUDING
(01014) *          A(8),(T(J),OUT(J),EXPN(J),+B(J),-B(J)) ( IN SEQUENCE )
(01015) *
(01016) *       * RESTRICTION:
(01017) *
```

```
                    (01018) *       1. THE ORDER OF THE PREDICTOR IS 4.
                    (01019) *       2. THE LOCATIONS OF SOME ARRAYS ARE AS FOLLOWS:
                    (01020) *          SAMPLE BUFFER: 0(3) ... 1023(3)
                    (01021) *          VHAT BUFFER: 0(2) ... 1023(2)
                    (01022) *          SHAT BUFFER: 1024(3) ... 2047(3)
                    (01023) *          PARAMETER BUFFER: 3072(2)  ...
                    (01024) *       3. THE SEQUENCE OF INPUT SCALARS ARE AS FOLLOWS:
                    (01025) *          BITCOUNT,BIT BUFFER SIZE,RMS,STATE VARIABLE,RUN LENGTH,
                    (01026) *          PHONY BETA BITS,GAP SIZE,RMSMIN,N-2.5,1-ALP,ALP,SMIN,
                    (01027) *          G,NQ,RUN LENGTH,COMPANSATOR,2**-14,AINV*(1/ALAD-1),
                    (01028) *          ALAD,N-2.5,NULL BITS,1/2**15,NSTX,ENCD.T,ENCD.BETA,
                    (01029) *          BETA,ENCD.T,ENCD.BETA,BETA
                    (01030) *
                    (01031) *       * TO CALL THIS FUNCTION:
                    (01032) *
                    (01033) *         MAP = PCTX(SA,U)
                    (01034) *         WHERE  SA --- THE SCALAR LOCATION FOR BETA
                    (01035) *                U  --- THE BUFFER # FOR OUTPUT QUANTIZING LEVELS
                    (01036) *
                    (01037) *
                    (01038) *       DISPATCH TABLE ENTRY
                    (01039) *
          000000F2  (01040)         FCB242=242                      ;FCB=242
          00000B94  (01041)         #L=AFDT$+3*2*(FCB242-128)       ;STARTING LOCATION OF AFDT FOR 242
                    (01042) *
    0B94  01E4AFA   (01043)         ADDR      PCTX$(R7,1)           ;APU=PCTX
    0B96  01E4C84   (01044)         ADDR      VPCTX$(R7,1)          ;APS=VPCTX
    0B98  01021FC   (01045)         ADDR      CSPU$NOS(,1,0)        ;NO CSPU SUPPORT
                    (01046) *
                    (01047) *       APU & APS MODULE
                    (01048) *
          00004AF8  (01049)         #L=CON2$
                    (01050) *
                    (01051) *       APU MODULE
                    (01052) *
                    (01053)         EVEN                            ;START ON WORD BOUNDARY
    4AF8  0000      (01054)         DATA      PCTX$SA               ;STARTING ADDRESS
    4AF9  00C1      (01055)         DATA      PCTX$SZ               ;SIZE
                    (01056) *
                    (01057) PCTX$   BEGIN     APU(PCTX)             ;START OF APU MODULE
          00000003  (01058)         #M=3
          00000000  (01059)         #A=0
                    (01060) *
                    (01061) *       INITIALIZATION
```

```
                    (01062) *
                    (01063) *       REGISTERS WHICH ARE PERMANENTLY USED IN THIS PROGRAM
                    (01064) *
                    (01065) *       ADM1: A2,A7
                    (01066) *       ADM2: A0,A2,A4,A5,A7
                    (01067) *
                    (01068) *       REGISTER AFFECTED:
                    (01069) *       ADM1: A2,A3
                    (01070) *       ADM2: A0,A2,A4,A5,A7,EXO
                    (01071) *
                    (01072) *       INPUT SEQUENCE: BITCOUNT,BIT BUFFER SIZE,RMS,STATE VARIABLE,RUN LENGTH
                    (01073) *
                    (01074) *       FUNCTIONS: NEW BITCOUNT = OLD BITCOUNT + BIT BUFFER SIZE
                    (01075) *
                    (01076) *                                          ADM1                ADM2
                    (01077) *                                          ****************    ****************
                    (01078) *
A00 04AFA 00F2168 (01079) PCTX$SA MOV(IQA,A2) \ K(1)              ;A2=BITCOUNT
A01 04AFC 00F3C814 (01080)        MOV(IQA,A3) \ MOV(ZERO,A4)      ;A3=BIT BUFFER SIZE   A4=SAMPLE COUNT
A02 04AFE 434000F2 (01081)        ADD(A2,A3) \ MOV(IQA,A2)        ;                     A2=RMS
A03 04B00 000008F5 (01082)        NOP \ MOV(IQA,A5)               ;-----                A5=STATE VARIABLE
A04 04B02 000008F0 (01083)        NOP \ MOV(IQA,A0)               ;                     A0=RUN LENGTH
A05 04B04 00F70897 (01084)        MOV(IQA,A7) \ MOV(R,A7)         ;                     A7=1
A06 04B06 00920098 (01085)        MOV(R,A2) \ MOV(R,EXO)          ;                     EXO=NO PHONY BETA
                    (01086) *
                    (01087) *       CHECK PITCH REPETITION MODE
                    (01088) *
                    (01089) *       REGISTERS USED:
                    (01090) *       ADM1: A2
                    (01091) *       ADM2: A7
                    (01092) *
                    (01093) *       REGISTERS AFFECTED:
                    (01094) *       ADM1: A1,A2
                    (01095) *       ADM2: A6
                    (01096) *
                    (01097) *       FLAGS SENSED: G2
                    (01098) *
                    (01099) *       INPUT SEQUENCE: PHONY BETA BIT,PITCH REPETITION SIZE,INPUT PITCH REPETIT
                    (01100) *       OUTPUT SEQUENCE: PHONY BETA CODE,OUTPUT PITCH REPETITION DATA
                    (01101) *
                    (01102) *       FUNCTIONS: BITCOUNT = BITCOUNT - PHONY BETA BITS
                    (01103) *                  PITCH REPETITION
                    (01104) *
A07 04B08 90006001 (01105)        JUMPC(PRE3-1,G2)                ;JUMP IF NO PITCH REPETITION
```

```
A80 0480A 80F180AE0 (0110)          MOV(IQA,A1) \ NEG(A7)          ;A1=PHONY BETA BITS
A89 0480C 49400000 (0107)           SUB(A2,A1) \ NOP               ;UPDATE BIT COUNT
A8A 0480E 00000F6 (0108)            NOP \ MOV(IQA,A6)              ;                        A6=PITCH REPETITION
A8B 04810 0892089C (0109)           MOV(R,A2) \ MOV(R,OQ)          ;                        OUTPUT PHONY BETA
A8C 04812 80FC4FC0 (0110) GAP1      MOV(IQA,OQ) \ SUB(A6,A7)       ;PITCH REPETITION
A8D 04814 00000896 (0111)           NOP \ MOV(R,A6)                ;
A8E 04816 981F00C (0112)            JUMPC(GAP1,T2)                 ;
A8F 04818 1000011 (0113)            JUMP(PRE3)
                    (0114) *
                    (0115) *        PREDICTOR
                    (0116) *
                    (0117) *        REGISTERS USED:
                    (0118) *        ADM1: A7
                    (0119) *        ADM2: A2,A5
                    (0120) *
                    (0121) *        REGISTERS AFFECTED:
                    (0122) *
                    (0123) *        ADM1: A1,A5,A7,M2,M7
                    (0124) *        ADM2: A1,A2,A3,A6,M0,M1,M6,M7,EXO
                    (0125) *
                    (0126) *        FLAGS AFFECTED: AF0,G1
                    (0127) *
                    (0128) *        INPUT SEQUENCE: RMSMIN,N-2.5,1-ALP,ALP,A(I),VHAT(K-I)
                    (0129) *
                    (0130) *        FUNCTIONS: P(K) = SUM(A(I) * VHAT(K-I))
                    (0131) *                   RMS = ALP*(RMS-RMSMIN) + (1-ALP)*ABS(VHAT(K-1)) + RMSMIN
                    (0132) *                   RMS ** 2
                    (0133) *
                    (0134) *
A10 0481A 085C000 (0135)            MOV(EXI,OQ) \ NOP              ;OUTPUT PHONY BETA
A11 0481C 2040204 (0136) PRE3       SET(AF0)                       ;SIGNAL APS --- PARC LOOP
A12 0481E 1600F6 (0137)             K(1) \ MOV(IQA,A6)             ;                        A6=RMSMIN
A13 04820 000003 (0138)             NOP \ MOV(IQA,A3)              ;                        A3=N-2.5
A14 04822 000008 (0139)             NOP \ MOV(IQA,M0)              ;                        M0=1-ALP
A15 04824 49F180EE (0140)           MOV(A1),SUB(A7,A1) \ MOV(IQA,M6) ;                      M6=ALP
                    (0141) *
A16 04826 08EA000 (0142)            MOV(IQA,M2) \ NOP              ;M2=A(1)
A17 04828 08CF000 (0143)            MOV(IQ,M7) \ NOP               ;M7=VHAT(K-1)            ------
A18 0482A 856008F1 (0144)           MUL(M2,M7) \ MOV(IQA,A1)       ;P1=A(1)*VHAT(K-1)       A1=VHAT(K-1)
A19 0482C 0815122 (0145)            MOV(ZERO,A5) \ ABS(A1)         ;A5=0.                   R=ABS(A1)
A1A 0482E 0897000 (0146)            MOV(R,A7) \ NOP                ;DECRE. Q COUNTER
A1B 04830 981E001D (0147)           JUMPC(QCKP,T1)
A1C 04832 2045205 (0148)            SET(G1)                        ;SIGNAL Q-LIMIT REACHED
A1D 04834 000008F (0149) QCKP       NOP \ MOV(R,M7)                ;------                  M7=ABS(VHAT(K-1))
```

```
A1E 0483 0 0B1846 0 (0115 0)          MOV(P,A1) \ MUL(M0,M7)          ;A1=P1                  P2=(1-ALP)*M7
A1F 0483 8 41A04E40 (0115 1)          ADD(A5,A1) \ SUB(A2,A6)         ;R1=PRE+P1              R2=RMS-RMSMIN
A20 0483 A EA0000 (0115 2)            MOV(IQA,M2) \ NOP               ;M2=A(2)                ------
A21 0483 C 0EF000 (0115 3)            MOV(IQA,M7) \ NOP               ;M7=VHAT(K-2)           ------
A22 0483 E 056009 (0115 4)            MUL(M2,M7) \ MOV(R,M1)          ;P3=A(2)*VHAT(K-2)      M1=R2
A23 0484 0 095D4D1 (0115 5)           MOV(R,A5) \ MOV(A1),MUL(M1,M6)  ;------                 P4=R2*ALP
A24 0484 2 0462 0 (0115 6)            NOP \ ADD(A1,A6)                ;------                 R3=P2+RMSMIN
A25 0484 4 0B1 0B2 (0115 7)           MOV(P,A1) \ MOV(P,A2)           ;A1=P3                  A2=P4
A26 0484 6 41A04231 (0115 8)          ADD(A5,A1) \ MOV(A1),ADD(A1,A2) ;EXO=NO-0.5             A1=R3
A27 0484 8 095008F (0115 9)           MOV(R,A5) \ MOV(R,M7)           ;A5=NEW PRE             M7=NEW RMS
A28 0484 A 00089 (0116 0)             NOP \ MOV(R,M1)                 ;------                 M1=NEW RMS
A29 0484 C 00892 (0116 1)             NOP \ MOV(R,A2)                 ;------                 A2=NEW RMS
                      (0116 2) *
                      (0116 3) *      PREDICTOR LOOP
                      (0116 4) *
A2A 0484 E 0EA04E0 (0116 5) PRE2      MOV(IQA,M2) \ MUL(M1,M7)        ;M2=A(I)                P=RMS**2
A2B 0485 0 0EF02A0 (0116 6)           MOV(IQA,M7) \ R(A5)             ;M7=VHAT(K-I)           R=SIZE
A2C 0485 2 056000 (0116 7)            MUL(M2,M7) \ NOP                ;P=A(I)*VHAT(K-I)       ------
A2D 0485 4 0B10B81 (0116 8)           MOV(P,A1) \ MOV(P,A1)           ;A1=A(I)*VHAT(K-I)      A1=RMS**2
A2E 0485 6 41A04F78 (0116 9)          ADD(A5,A1) \ MOV(EXO),SUB(A3,A7) ;R=NEW PRE             EXO=SIZE
A2F 0485 8 0895093 (0117 0)           MOV(R,A5) \ MOV(R,A3)           ;A5=RENEW PRE           A3=COUNTER-1
A30 0485 A 91F02A (0117 1)            JUMPC(PRE2,T2)                  ;NED?
                      (0117 2) *
                      (0117 3) *      RECIPROCAL PART
                      (0117 4) *
                      (0117 5) *      REGISTERS USED:
                      (0117 6) *      ADM1:
                      (0117 7) *      ADM2: A1,EXO
                      (0117 8) *
                      (0117 9) *      REGISTERS AFFECTED:
                      (0118 0) *
                      (0118 1) *      ADM1: A1,A3,M2,M6,M7
                      (0118 2) *      ADM2: A1,A3,M1,M6,M7
                      (0118 3) *
                      (0118 4) *      FUNCTIONS: 1/SIZE
                      (0118 5) *                 1/(RMS**2)
                      (0118 6) *
A31 0485 C 0851 0AF (0118 7)          MOV(EXI,A1) \ MOV(P,M7)         ;A1=SIZE                M7=RMS**2
A32 0485 E 2E202E20 (0118 8)          RCP(A1) \ RCP(A1)               ; R=1/SIZE+DEL          R=/RMS**2+DEL
A33 0486 0 084F000 (0118 9)           MOV(EXI,M7) \ NOP               ;M7=SIZE
A34 0486 2 089A0089 (0119 0)          MOV(R,M2) \ MOV(R,M1)           ;M2=F0                  M1=F0
A35 0486 4 056004E0 (0119 1)          MUL(M2,M7) \ MUL(M1,M7)         ;P1=F0*SIZE             P2=F0*RMS**2
A36 0486 6 16A016A0 (0119 2)          K(2) \ K(2)
A37 0486 8 0B30B83 (0119 3)           MOV(P,A3) \ MOV(P,A3)           ;A3=P1                  A3=P2
```

```
A38 04B6A 4B314B31 (01194)        MOV(A1),SUB(A1,A3) \ MOV(A1),SUB(A1,A3) ;R=2-P1      R=2-P2
A39 04B6C 008E008E (01195)        MOV(R,M6) \ MOV(R,M6)           ;M6=2-P1            M6=2-P2
A3A 04B6E 0540084C (01196)        MUL(M2,M6) \ MUL(M1,M6)         ;P=F0*(2-P1)        P=F0*(2-P2)
A3B 04B70 30000AAF (01197)        CALL(RCP1)                      ;CALL SUBROUTINE RCP1
A3C 04B72 30000AAF (01198)        CALL(RCP1)
                   (01199) *
                   (01200) *      CALCULATE REDUCED SPEECH
                   (01201) *
                   (01202) *      REGISTERS USED:
                   (01203) *      ADM1:
                   (01204) *      ADM2: A4,A7
                   (01205) *
                   (01206) *      REGISTERS AFFECTED:
                   (01207) *
                   (01208) *      ADM1: A0,A1,A4,A6,M1,M5,M6,M7
                   (01209) *      ADM2: A1,A4,A6,M0,M4,M7
                   (01210) *
                   (01211) *      INPUT SEQUENCE: SHAT(K-T),BETA,SMIN,G,NQ,RUNCOUNT,COMPANSATION,S(K),1/2*
                   (01212) *
                   (01213) *      FUNCTIONS: V(K) = S(K) - BETA * SHAT(K-T)
                   (01214) *                 SAMPLE COUNT = SAMPLE COUNT + 1
                   (01215) *
A3D 04B74 008EFAAA (01216)        MOV(IQA,M7) \ NOP               ;M7=SHAT(K-T)
A3E 04B76 008E9AAA (01217)        MOV(IQA,M1) \ NOP               ;M1=BETA
A3F 04B78 04EE478A (01218)        MOV(M6),MUL(M1,M7) \ ADD(A4,A7) ;M6=1/SIZE
A40 04B7A 168A008E8 (01219)       K(1) \ MOV(IQA,M0)              ;                   M0=SMIN
A41 04B7C 0AAAA08EF (01220)       NOP \ MOV(IQA,M7)               ;                   M7=G
A42 04B7E 008F4A08F1 (01221)      MOV(IQA,A4) \ MOV(IQA,A1)       ;A4=NQ              A1=NQ
A43 04B80 0AAAA8F6 (01222)        NOP \ MOV(IQA,A6)               ;                   A6=RUNLENGTH
A44 04B82 008FAAAA (01223)        MOV(IQA,A0) \ NOP               ;A0=COMPANSATION
A45 04B84 008F1AA0 (01224)        MOV(IQA,A1) \ NOP               ;A1=S(K)
A46 04B86 00B6008C (01225)        MOV(P,A6) \ MOV(R,M4)           ;A6=BETA*SHAT(K-T)
A47 04B88 4E290894 (01226)        MOV(M1),SUB(A1,A6) \ MOV(R,A4) ;R=S(K)-BETA*SHAT(K-T) A4=SAMPLE COUNT
A48 04B8A 00ED0AAA (01227)        MOV(IQA,M5) \ NOP               ;M5=1/2**14
                   (01228) *
                   (01229) *      QUANTIZER
                   (01230) *
                   (01231) *      REGISTERS USED:
                   (01232) *      ADM1: A0,A2,A4,A5,M1,M5,M6
                   (01233) *      ADM2: A1,A2,A5,A6,A7,M0,M7
                   (01234) *
                   (01235) *      REGISTERS AFFECTED:
                   (01236) *
                   (01237) *      ADM1: A1,A2,A3,A4,M1,M2,M7,EXO
```

```
                    (0)1238) *       ADM2: A0,A1,A3,A5,M1,M6,M7,EXO
                    (0)1239) *
                    (0)1240) *       INPUT SEQUENCE: T(J),OUT(J),EXPN(J),B(J)
                    (0)1241) *       OUTPUT SEQUENCE: VHAT(K),L
                    (0)1242) *
                    (0)1243) *       FLAGS AFFECTED: G0,AF0,AF1,AF2,AF3
                    (0)1244) *       FLAGS SENSED: G0,AF0,AF3
                    (0)1245) *
                    (0)1246) *       FUNCTIONS: E(K) = V(K) - P(K)
                    (0)1247) *                  ABS(E(K)/SIZE) C.F. T(J)
                    (0)1248) *                  RMS * SMIN
                    (0)1249) *                  G/(RMS**2)
                    (0)1250) *                  GET QUANTIZER LEVEL
                    (0)1251) *                  SIZE * EXPN(J)
                    (0)1252) *                  EHAT(K) = +- OUT(J) * SIZE
                    (0)1253) *                  VHAT(K) = EHAT(K) + P(K)
                    (0)1254) *                  SIZE = MAX(SIZE * EXPN(J) , RMS * SMIN)
                    (0)1255) *                  OUTPUT VHAT(K) AND QUANTIZER LEVEL
                    (0)1256) *                  CHECK RUN LENGTH CONDITION
                    (0)1257) *                  UPDATE RUN LENGTH COUNTER AND BIT COUNTER
                    (0)1258) *
A49 04B0C 0091024D (0)1259)         MOV(R,A1) \ R(A2)              ;A1=V(K)              R=RMS
A4A 04B0E 4D200000 (0)1260)         SUB(A1,A5) \ NOP               ;R1=V(K)-PRE
A4B 04B0F 008A08A9 (0)1261)         MOV(R,M2) \ MOV(P,M1)          ;M2=R1                M1=1/RMS**2
A4C 04B02 901E004E (0)1262)         JUMPC(CONT2,T1)                ;JUMP IF ERROR > 0
A4D 04B04 20044204 (0)1263)         SET(G0)                        ;SET FLAG G0
A4E 04B06 0000008E (0)1264) CONT2   NOP \ MOV(R,M6)                ;-----                M6=RMS
A4F 04B08 B5408440 (0)1265)         ABSMUL(M2,M6) \ MUL(M0,M6)     ;P=ABS(R1/SIZE)       P=RMS*SMIN
A50 04B0A 00B10083 (0)1266)         MOV(P,A1) \ MOV(P,A3)          ;A1=ABS(R1/SIZE)      A3=RMS*SMIN
A51 04B0C 000084E0 (0)1267)         NOP \ MUL(M1,M7)               ;------              P1=G*1/RMS**2
                    (0)1268) *
                    (0)1269) *       QUANTIZER LOOP
                    (0)1270) *
                    (0)1271) *
A52 04B0E 00000000 (0)1272) QUAN1   NOP                            ;
A53 04B0F 91000052 (0)1273)         JUMPS(QUAN1,AF0)               ;
A54 04B02 20402048 (0)1274)         SET(AF0)                       ;SIGNAL APS TO PRODUCE T(K) ADDRESS
A55 04B04 00F30000 (0)1275)         MOV(IQA,A3) \ NOP              ;A3=T(K)              ------
A56 04B06 6320402F (0)1276)         MAX(A1,A3) \ SUB(A1,A7)        ;ERROR > T(K)?        R=COUNTER-1
A57 04B08 00000091 (0)1277)         MOV(R,NULL) \ MOV(R,A1)        ;WAIT FOR T1 FLAG     A1=COUNTER-1
A58 04B0A 911E00BF (0)1278)         JUMPS(QUAN6,T1)                ;ERROR < T(K) GO TO QUAN6
A59 04B0C 20402048 (0)1279)         SET(AF3)                       ;L=1
A5A 04B0E 901F0052 (0)1280)         JUMPC(QUAN1,T2)                ;COUNTER > 0 GO TO QUAN1
A5B 04B00 20402040 (0)1281)         SET(AF2)                       ;SIGNAL APS TO STOP PRODUCING T(K) ADDRES
```

```
A5C 04BB2 0000F20 (01282)          NOP \ SUB(A1,A7)                ;A1-1
A5D 04BB4 000028B (01283) QUAN2    NOP \ MOV(EXO),R(A5)
A5E 04BB6 0851000 (01284)          MOV(EXI,A1) \ NOP
A5F 04BB8 0000898 (01285)          NOP \ MOV(R,EXO)                ;CHECK ERROR          EXO=STATE VARIABLE
A60 04BBA 9104007 (01286)          JUMPS(NEGA,G0)                  ;IF ERROR<0 GO TO NEGA
                  (01287) *
                  (01288) *        ERROR >= 0
                  (01289) *
A61 04BBC 004A08E9 (01290)         MOV(EXI,M2) \ MOV(IQA,M1)       ;M2=STATE VARIABLE    M1=OUT(J)
A62 04BBE 00EF08E (01291)          MOV(IQA,M7) \ MOV(R,M6)         ;M7=EXPN(J)           M6=STATE VARABLE(SI
A63 04BC0 856084CF (01292)         MUL(M2,M7) \ MOV(M7),MUL(M1,M6) ;SIZE*EXPN(J)         M7=P1 P=OUT(J)*SIZE
A64 04BC2 00B03B8 (01293)          MOV(P,EXO) \ MOV(P,EXO)         ;EXO=SIZE*EXPN(J)     EXO=OUT(J)*SIZE
A65 04BC4 0853081 (01294) QUAN3    MOV(EXI,A3) \ MOV(EXI,A1)       ;A3=+-OUT(J)*SIZE     A1=SIZE*EXPN(J)
A66 04BC6 04A0000 (01295)          MUL(M1,M5) \ NOP                ;P=1/2**14
A67 04BC8 4560632 (01296)          ADD(A3,A5) \ MAX(A1,A3)         ;OUT(J)*SIZE+PRE      CHOOSE NEW SIZE
A68 04BCA 0858095 (01297)          MOV(EXI,EXO) \ MOV(R,A5)        ;EXO=+-OUT(J)*SIZE    RENEW SIZE
A69 04BCC 0893049 (01298)          MOV(R,A3) \ MOV(EXI,M1)         ;A3=VHAT(K)           M1=EQ
                  (01299) *
A6A 04BCE 009C000 (01300)          MOV(R,OQ) \ NOP                 ;OUTPUT VHAT(K)
A6B 04BD0 9000075 (01301)          JUMPC(QUAN4,AF3)                ;JUMP IF Q(L)=1
A6C 04BD2 498002C8 (01302)         SUB(A4,A1) \ R(A6)              ;CALCULATE L          A6=RUNLENGTH
A6D 04BD4 2020202B (01303)         CLEAR(AF3)                      ;CLEAR AF3
A6E 04BD6 00A0000 (01304)          MOV(P,NULL) \ NOP               ;
A6F 04BD8 0080000 (01305)          MOV(R,M1) \ NOP                 ;M1=L
A70 04BDA 04A0000 (01306)          MUL(M1,M5) \ NOP                ;READY TO ALIGN
A71 04BDC 0084000 (01307)          MOV(P,A4) \ NOP                 ;
A72 04BDE 3A00000 (01308)          ALIGN(A4) \ NOP                 ;
A73 04BE0 009C009 (01309)          MOV(R,OQ) \ MOV(R,A0)           ;OUTPUT L             A0=RUNCOUNT
A74 04BE2 1000078 (01310)          JUMP(QUAN5)                     ;
A75 04BE4 0044F00 (01311) QUAN4    MOV(P,A4) \ SUB(A0,A7)          ;                     OUTPUT L=1
A76 04BE6 3A00000 (01312)          ALIGN(A4) \ NOP                 ;
A77 04BE8 009C009 (01313)          MOV(R,OQ) \ MOV(R,A0)           ;                     DECREMENT A0
A78 04BEA 901F078 (01314)          JUMPC(QUAN5,T2)                 ;JUMP IF NO ENOUGH FOR RUN LENGTH
A79 04BEC 4200020 (01315)          ADD(A0,A2) \ R(A6)              ;COMPASATING          A6=RUNLENGTH
A7A 04BEE 0092009 (01316)          MOV(R,A2) \ MOV(R,A0)           ;                     A0=RUNCOUNT
A7B 04BF0 00F1000 (01317) QUAN5    MOV(IQA,A1) \ NOP               ;A1=B(K)
A7C 04BF2 4940000 (01318)          SUB(A2,A1) \ NOP                ;UPDATE BITCOUNT
A7D 04BF4 0092000 (01319)          MOV(R,A2) \ NOP                 ;
A7E 04BF6 901E000 (01320)          JUMPC(ADAP1,T1)                 ;JUMP IF BITCOUNT > 0
A7F 04BF8 2040240 (01321)          SET(AF3)                        ;SIGNAL END OF PARC
                  (01322) *
                  (01323) *        ADAPTATION
                  (01324) *
                  (01325) *        REGISTERS USED:
```

```
                    (0001326) *       ADM1: A3,A6
                    (0001327) *       ADM2: A7,M1,M7
                    (0001328) *
                    (0001329) *       REGISTERS AFFECTED:
                    (0001330) *
                    (0001331) *       ADM1: A0,A1,A4,A5,A6,M1,M2,M6
                    (0001332) *       ADM2: A3,M1,M6,M7,EXO
                    (0001333) *
                    (0001334) *       INPUT SEQUENCE: AINV*(1/ALAD-1),ALAD,N-2.5,NULL BITS,
                    (0001335) *                       A(I),VHAT(K-I)
                    (0001336) *       OUTPUT SEQUENCE: SHAT(K),A(I)
                    (0001337) *
                    (0001338) *       FLAGS AFFECTED: G0
                    (0001339) *
                    (0001340) *       FUNCTIONS: ERR = G * EHAT(K) / (RMS ** 2)
                    (0001341) *                  A(1) = (A(1) + AINV*(1/ALAD-1)) * ALAD + VHAT(K-1)*ERR
                    (0001342) *                  SHAT(K) = VHAT(K) + BETA*SHAT(K-T)
                    (0001343) *                  OUTPUT SHAT(K) AND A(1)
                    (0001344) *                  A(I) = A(I)*ALAD + VHAT(K-I)*ERR
                    (0001345) *                  OUTPUT A(I)
                    (0001346) *
A00 04BFA 00F40000  (0001347) ADAP1   MOV(IQA,A4) \ NOP              ;A4=AINV*(1/ALAD-1)
A01 04BFC 00EE0000  (0001348)         MOV(IQA,M6) \ NOP              ;M6=ALAD
A02 04BFE 000000F3  (0001349)         NOP \ MOV(IQA,A3)              ;                        A3=N-2.5
A03 04C00 00F00000  (0001350)         MOV(IQA,A0) \ NOP              ;A0=NULL BIT
                    (0001351) *
A04 04C02 00F584E0  (0001352)         MOV(IQA,A5) \ MUL(M1,M7)       ;A5=A(1)                 P1=G*EQ/RMS**2
A05 04C04 44A00000  (0001353)         ADD(A5,A4) \ NOP               ;A(1)+A4
A06 04C06 466A00A9  (0001354)         MOV(M2),ADD(A3,A6) \ MOV(P,M1) ;M2=R1                   M1=P1
A07 04C08 054000EF  (0001355)         MUL(M2,M6) \ MOV(IQA,M7)       ;P2=ALAD*R1              M7=VHAT(K-1)
A08 04C0A 00E984E0  (0001356)         MOV(IQA,M1) \ MUL(M1,M7)       ;M1=A(2)                 P3=VHAT(K-1)*P1
A09 04C0C 009C0000  (0001357)         MOV(R,OQ) \ NOP                ;OUTPUT SHAT(K)
A0A 04C0E 04D10000  (0001358)         MOV(A1),MUL(M1,M6) \ MOV(P,EXO) ;A1=P2                  EXO=P3
A0B 04C10 00560EE   (0001359)         MOV(EXI,A6) \ MOV(IQA,M6)      ;A6=VHAT(K-1)*ERR        M6=VHAT(K-2)
A0C 04C12 41C004C0  (0001360)         ADD(A6,A1) \ MUL(M1,M6)        ;R=NEW A(1)              P=VHAT(K-2)*ERR
A0D 04C14 20242024  (0001361)         CLEAR(G0)                      ;
A0E 04C16 009C0000  (0001362)         MOV(R,OQ) \ NOP                ;OUTPUT A(1)
A0F 04C18 00010000  (0001363)         MOV(P,A1) \ MOV(P,EXO)         ;A1=ALAD*A(2)            EXO=VHAT(K-2)*ERR
A00 04C1A 00560000  (0001364)         MOV(EXI,A6) \ NOP              ;A6=VHAT(K-2)*ERR
A01 04C1C 41C00000  (0001365)         ADD(A6,A1) \ NOP               ;R=NEW A(2)
A02 04C1E 009C0000  (0001366)         MOV(R,OQ) \ NOP                ;OUTPUT A(2)
A03 04C20 00E90000  (0001367) ADAP2   MOV(IQA,M1) \ NOP              ;M1=A(I)
A04 04C22 04C00EF   (0001368)         MUL(M1,M6) \ MOV(IQA,M7)       ;P=A(I)*ALAD             M7=VHAT(K-I)
A05 04C24 0000B4E0  (0001369)         NOP \ MUL(M1,M7)               ;                        P=VHAT(K-I)*ERR
```

```
A96 04C26 08B10808 (01370)          MOV(P,A1) \ MOV(P,EXO)          ;A1=A(I)*ALAD          EXO=VHAT(K-I)*ERR
A97 04C28 08564F60 (01371)          MOV(EXI,A6) \ SUB(A3,A7)        ;A6=VHAT(K-I)*ALAD     COUNTER=COUNTER-1
A98 04C2A 41C00000 (01372)          ADD(A6,A1) \ NOP                ;R=NEW A(I)
A99 04C2C 0809C803 (01373)          MOV(R,OQ) \ MOV(R,A3)           ;OUTPUT NEW A(I)
A9A 04C2E 981F0093 (01374)          JUMPC(ADAP2,T2)                 ;END ?
A9B 04C30 00000000 (01375)          NOP
                   (01376) *
                   (01377) *        END OF PROCESSING ?
                   (01378) *
                   (01379) *        FLAGS SENSED: G1,AF3
                   (01380) *
A9C 04C32 91000A02 (01381)          JUMPS(FEND,AF3)                 ;JUMP IF BITCOUNT < 0
A9D 04C34 90050011 (01382)          JUMPC(PRE3,G1)                  ;JUMP IF NOT RUN OUT OF SAMPLE
                   (01383) *
                   (01384) *        NULL MODE
                   (01385) *
                   (01386) *        REGISTERS USED:
                   (01387) *        ADM1: A0,A2
                   (01388) *        ADM2: A2,A5,A7,M4
                   (01389) *
                   (01390) *        REGISTERS AFFECTED:
                   (01391) *
                   (01392) *        ADM1: A2
                   (01393) *        ADM2: M3,EXO
                   (01394) *
                   (01395) *        FLAGS AFFECTED: G1,AF3,RA
                   (01396) *
                   (01397) *        INPUT SEQUENCE: 1/2**15
                   (01398) *        OUTPUT SEQUENCE: END CODE,BIT COUNT,RMS,STATE VARIABLE,# OF SAMPLES
                   (01399) *
                   (01400) *        FUNCTIONS: IF MORE THAN TWO NULL CODES, SET BIT COUNT = 0
                   (01401) *                   OTHERWISE UPDATE BITCOUNT
                   (01402) *                   OUTPUT END CODE, BITCOUNT, RMS, SIZE AND SAMPLE COUNTER
                   (01403) *
A9E 04C36 48400000 (01404) NULL1    SUB(A2,A0) \ NOP                ;UPDATE BITCOUNT
A9F 04C38 08020005 (01405)          MOV(R,A2) \ NOP                 ;
AA0 04C3A 911E00A2 (01406)          JUMPS(FEND,T1)                  ;JUMP IF BITCOUNT < 0
AA1 04C3C 08120000 (01407)          MOV(ZERO,A2) \ NOP              ;RESET BITCOUNT TO SYNC. TRANSMISSIONS
AA2 04C3E 000000EB (01408) FEND     NOP \ MOV(IQA,M3)               ;                      M3=1./(2.**15)
AA3 04C40 00005800 (01409)          NOP \ MUL(M3,M4)                ;
AA4 04C42 02400AE0 (01410)          R(A2) \ NEG(A7)                 ;
AA5 04C44 0000025C (01411)          NOP \ MOV(OQ),R(A2)             ;                      OUTPUT END CODE
AA6 04C46 0809C808 (01412)          MOV(R,OQ) \ MOV(P,EXO)          ;OUTPUT BITCOUNT
AA7 04C48 0020202C (01413)          MOV(EXI,A2) \ MOV(OQ),R(A5)     ;                      OUTPUT RMS
```

AD-A086 134 NOTRE DAME UNIV IN DEPT OF ELECTRICAL ENGINEERING F/G 17/2
DESIGN AND IMPLEMENTATION OF A SPEECH CODING ALGORITHM AT 9600 --ETC(U)
APR 80 J L MELSA, D L COHN, A ARORA DCA100-79-C-0005

UNCLASSIFIED NL

3 OF 3
AD
A086134

END
DATE
FILMED
8-80
DTIC

```
AA8 04C4A 3A40009C (01414)          ALIGN(A2) \ MOV(R,OQ)          ;                          OUTPUT STATE VARIAB
AA9 04C4C 009C0000 (01415)          MOV(R,OQ) \ NOP                ;OUTPUT # OF SAMPLES
AAA 04C4E 20202020 (01416)          CLEAR(AF3)                     ;
AAB 04C50 20252025 (01417)          CLEAR(G1)                      ;
AAC 04C52 20322032 (01418)          CLEAR(RA)                      ;
AAD 04C54 00000000 (01419)          NOP
AAE 04C56 10000000 (01420)          JUMP(0)
                   (01421) *
AAF 04C58 00AA00A9 (01422) RCP1     MOV(P,M2) \ MOV(P,M1)          ;M2=F1                     M1=F1
AB0 04C5A 0560046E (01423)          MUL(M2,M7) \ MUL(M1,M7)        ;P=F1*SIZE                 P=F1*RMS**2
AB1 04C5C 00B300B3 (01424)          MOV(P,A3) \ MOV(P,A3)          ;A3=F1*SIZE                A3=F1*RMS**2
AB2 04C5E 4B204B20 (01425)          SUB(A1,A3) \ SUB(A1,A3)        ;R6=2-F1*SIZE              R7=2-F1*RMS**2
AB3 04C60 00800080 (01426)          MOV(R,M6) \ MOV(R,M6)          ;M6=R6                     M6=R7
AB4 04C62 0540046C (01427)          MUL(M2,M6) \ MUL(M1,M6)        ;P=F2                      P=F2
AB5 04C64 A000A000 (01428)          RETURN
AB6 04C66 00000000 (01429)          NOP
                   (01430) *
                   (01431) *        ERROR < 0
                   (01432) *
AB7 04C68 00040A00E9 (01433) NEGA   MOV(EXI,M2) \ MOV(IQA,M1)      ;M2=SIZE                   M1=OUT(J)
AB8 04C6A 00EF00E0 (01434)          MOV(IQA,M7) \ MOV(R,M6)        ;M7=EXPN(J)                M6=SIZE
AB9 04C6C 0560046CF (01435)         MUL(M2,M7) \ MOV(M7),MUL(M1,M6) ;P=SIZE*EXPN(J)       M3=P1, P=OUT(J)*SIZ
ABA 04C6E 4C200081 (01436)          SUB(A1,A4) \ MOV(P,A1)
ABB 04C70 00910A20 (01437)          MOV(R,A1) \ NEG(A1)            ;------                    R=-SIZE*OUT(J)
ABC 04C72 00B80098 (01438)          MOV(P,EXO) \ MOV(R,EXO)        ;EXO=SIZE*EXPN(J)          EXO=-OUT(J)*SIZE
ABD 04C74 10000065 (01439)          JUMP(QUAN3)
ABE 04C76 00000000 (01440)          NOP
ABF 04C78 20490049 (01441) QUAN6    SET(AF1)
AC0 04C7A 1000005D (01442)          JUMP(QUAN2)
                   (01443) *
          000000C1 (01444)          PCTX$SZ=*A-PCTX$SA             ;COMPUTE THE SIZE OF THIS MODULE
     04C7C        (01445)          END
                   (01446) *
                   (01447)          EJECT
```

```
                    (01448) *
                    (01449) *       APS MODULE OF PCTX PROGRAM
                    (01450) *
                    (01451)         EVEN                            ;START ON WORD BOUNDARY
    04C7C 0000D082  (01452)         ADDR    VPCTX$I                 ;PTR TO CONSTR INSTR BLOCK
    04C7E 0000CF4   (01453)         ADDR    VPCTX$+2*VPCTX$S        ;STARTING ADDR. OF SCALARS
    04C80 0001      (01454)         DATA    1                       ;ONE SCALAR
    04C81 0100      (01455)         DATA    VPCTX$Z                 ;SIZE
    04C82 0000C92   (01456)         ADDR    VPCTX$A                 ;CHAIN ANCHOR
                    (01457)         EVEN
                    (01458) *
                    (01459) VPCTX$  BEGIN   APS(VPCTX)
                    (01460) *
                    (01461) *       THE FOLLOWING REGISTERS ARE PERMENENTLY USED BY THIS PROGRAM
                    (01462) *
                    (01463) *       BR1: ADDRESS OF VHAT(K)
                    (01464) *       BR2: ADDRESS OF S(K-1)
                    (01465) *       BR3: ADDRESS OF SHAT(K-T-1)
                    (01466) *       BW0: 1023
                    (01467) *       BW1: ADDRESS OF SHAT(K-1)
                    (01468) *       BW2: SD(K+N-1)
                    (01469) *       BW3: QUANTIZER OUTPUT ADDRESS - 1
                    (01470) *
                    (01471) *       INITIALIZATION
                    (01472) *
                    (01473) *       REGISTERS AFFECTED: BR0,BR2,BW3
                    (01474) *
                    (01475) *       INPUT SEQUENCE: BIT COUNT, BIT BUFFER SIZE, RMS, SIZE, RUN LENGTH,
                    (01476) *                       PITCH REPETITION SIZE
                    (01477) *
  A00 04C84 00300032 (01478)         SET(RA)
                    (01479) *
  A01 04C86 02C204A4 (01480)         LOAD(BR0,SVT63$(1),L,TF)        ;BITCOUNT
  A02 04C88 048A0002 (01481)         ADD(BR0,2,TF)                   ;BIT BUFFER SIZE
  A03 04C8A 068A0002 (01482)         ADD(BR0,2,TF)                   ;RMS
  A04 04C8C 088A0002 (01483)         ADD(BR0,2,TF)                   ;STATE VARIABLE
  A05 04C8E 0A8A0002 (01484)         ADD(BR0,2,TF)                   ;RUN LENGTH
  A06 04C90 0CC2046E (01485)         LOAD(BR0,SVT118$(1),L,TF)       ;Q-BUFFER LIMIT
  A07 04C92 0E401000 (01486)         LOAD(BR0,[1])                   ;Q BUFFER
  A08 04C94 10220000 (01487)         SUB(BR2,M$$)
  A09 04C96 12020000 (01488)         SUB(BR0,M$$)                    ;
  A0A 04C98 14310011 (01489)         MOVB(BW3,BR0)                   ;ADDRESS POINTER OF Q BUFFER - 1
                    (01490) *
                    (01491) *       CHECK PITCH REPETITION
```

```
                    (0)1492) *
                    (0)1493) *       FLAG SENSED: G2
                    (0)1494) *
                    (0)1495) *       REGISTER AFFECTED: BR0,BR2,BR3,BW0,BW1,BW3
                    (0)1496) *
A0B 04C9A 1731A039 (0)1497) START   ADDL(BW3,1,TE)                  ; PHONY BETA ADDRESS
A0C 04C9C 1829A012 (0)1498)         MOVB(BR2,BW1)                  ;STARTING ADDRESS OF INPUT SPEECH(K-1)
A0D 04C9E 1A7E040A (0)1499)         LOAD(BR3,1024(3),S)            ;STARTING LOCATION OF SHAT BUFFER **
A0E 04CA0 1C39A02A (0)1500)         ADDB(BR3,BW1)                  ;
A0F 04CA2 1E11A017 (0)1501)         MOVB(BW1,BR3)                  ;RESERVE FOR PITCH REPETITION TO USE
A10 04CA4 2039A020 (0)1502)         SUBB(BR3,BW0)
A11 04CA6 224A03FF (0)1503)         LOAD(BR0,1023)                 ;SAMPLE BUFFER SIZE - 1  **
A12 04CA8 2401A011 (0)1504)         MOVB(BW0,BR0)
A13 04CAA 26A072A6 (0)1505)         JUMPC(VCHECK,G2)               ;JUMP IF NO PITCH REPETITION
A14 04CAC 28C204AE (0)1506)         LOAD(BR0,SVT68S(1),L,TF)       ;PHONY BETA BITS
A15 04CAE 2A0A0002 (0)1507)         ADD(BR0,2,TF)                  ;PITCH REPETITION SIZE
          00004CB1 (0)1508)         INSTX=@L+1                     ;INITIALIZE PITCH REPETITION SIZE
A16 04CB0 2C40003C (0)1509)         LOAD(BR0,60)                   ;PITCH REPETITION SIZE
A17 04CB2 2E11A021 (0)1510)         SUBB(BW1,BR0)                  ;BACK A PITCH REPETITION
A18 04CB4 3039A021 (0)1511)         SUBB(BR3,BR0)                  ;BACK A PITCH REPETITION
A19 04CB6 32020001 (0)1512)         SUB(BR0,1)                     ;PITCH REPETITION-1
A1A 04CB8 343A0001 (0)1513) VCAP    ADD(BR3,1)                     ;
A1B 04CBA 3639A000 (0)1514)         ANDB(BR3,BW0)                  ;
A1C 04CBC 38BA0400 (0)1515)         ADD(BR3,1024,TF)               ;SHAT(K-T-G) **
A1D 04CBE 3A11A039 (0)1516)         ADDL(BW1,1)                    ;
A1E 04CC0 3C11A000 (0)1517)         ANDB(BW1,BW0)                  ;RESTRRICT SHAT BUFFER TO START FROM 1024
A1F 04CC2 3E11A028 (0)1518)         ADDB(BW1,BW0)                  ;
A20 04CC4 40911A039 (0)1519)        ADDL(BW1,1,TF)                 ;
A21 04CC6 4209A1AB1 (0)1520)        SUBL(BR0,1),JUMPP(VCAP)        ;PITCH REPETITION
A22 04CC8 44A07260 (0)1521)         JUMP(VCHECK)                   ;
                    (0)1522) *
                    (0)1523) *       PREDICTOR
                    (0)1524) *
                    (0)1525) *       FLAGS AFFECTED: AF0
                    (0)1526) *
                    (0)1527) *       REGISTERS AFFECTED: BR0,BR1
                    (0)1528) *
A23 04CCA 46C20412 (0)1529) VLOOP   LOAD(BR0,SVT70S(1),L,TF)       ;RMSMIN
A24 04CCC 480A0002 (0)1530)         ADD(BR0,2,TF)                  ;N-2.5
A25 04CCE 4A0A0002 (0)1531)         ADD(BR0,2,TF)                  ;1-ALP
A26 04CD0 4C0A0002 (0)1532)         ADD(BR0,2,TF)                  ;ALP
A27 04CD2 4E200028 (0)1533)         CLEAR(AF0)                     ;
A28 04CD4 50C40C00 (0)1534)         LOAD(BR0,3072(2),L,TF)         ;STARTING LOCATION OF A(I) **
A29 04CD6 52120001 (0)1535)         SUB(BR1,1)                     ;
```

```
A2A 04CD8 5499####  (#1536)          ANDB(BR1,BW#,TF)
A2B 04CDA 6612###1  (#1537)          SUB(BR1,1)
A2C 04CDC 588A###2  (#1538)          ADD(BR#,2,TF)
A2D 04CDE 5A99####  (#1539)          ANDB(BR1,BW#,TF)
A2E 04CE# 5C12###1  (#154#)          SUB(BR1,1)
A2F 04CE2 5E8A###2  (#1541)          ADD(BR#,2,TF)
A3# 04CE4 6#99####  (#1542)          ANDB(BR1,BW#,TF)
A31 04CE6 6212###1  (#1543)          SUB(BR1,1)
A32 04CE8 648A###2  (#1544)          ADD(BR#,2,TF)
A33 04CEA 6699####  (#1545)          ANDB(BR1,BW#,TF)
A34 04CEC 681A###4  (#1546)          ADD(BR1,4)                        ;
                    (#1547) *
                    (#1548) *        PRODUCE REDUCED SPEECH AND CHECK UNDERFLOW OF SAMPLE BUFFER
                    (#1549) *
                    (#155#) *        REGISTERS AFFECTED: BR#,BR2,BR3
                    (#1551) *
                    (#1552) *        FLAGS AFFECTED: G1
                    (#1553) *
A35 04CEE 6A3A###1  (#1554)          ADD(BR3,1)
A36 04CF# 6C39####  (#1555)          ANDB(BR3,BW#)                     ;SHAT(K-T) ADDRESS
A37 04CF2 6EBA#4##  (#1556)          ADD(BR3,1#24,TF)
A38 04CF4 7#C2####  (#1557) VPCTX$S  LOAD(BR#,M$$(1),L,TF)             ;BETA
A39 04CF6 72C2#41A  (#1558)          LOAD(BR#,SVT74$(1),L,TF)          ;SMIN
A3A 04CF8 748A###2  (#1559)          ADD(BR#,2,TF)                     ;G
A3B 04CFA 768A###2  (#156#)          ADD(BR#,2,TF)                     ;NQ
A3C 04CFC 788A###2  (#1561)          ADD(BR#,2,TF)                     ;RUN LENGTH
A3D 04CFE 7A8A###2  (#1562)          ADD(BR#,2,TF)                     ;COMPANSATION
A3E 04D## 7C2A###1  (#1563)          ADD(BR2,1)
A3F 04D#2 7EA9####  (#1564)          ANDB(BR2,BW#,TF)                  ;S(K) ADDRESS
A4# 04D#4 8##A###2  (#1565)          ADD(BR#,2,TF)                     ;1/2**14
A41 04D#6 82#9##14  (#1566)          MOVB(BR#,BW2)                     ;
A42 04D#8 84#945E5  (#1567)          SUBB(BR#,BR2),JUMPN(VCONT1)       ;
A43 04D#A 86#94581  (#1568)          SUBL(BR#,1),JUMPP(VCONT1)         ;JUMP IF BR2 NOT EQUAL BW2-1
A44 04D#C 883##25   (#1569)          SET(G1)                           ;RUN OUT OF SAMPLE
                    (#157#) *
                    (#1571) *        PRODUCING ADDRESSES FOR QUANTIZER
                    (#1572) *
                    (#1573) *        FLAGS SENSED: G#,AF#,AF1,AF2
                    (#1574) *        FLAGS AFFECTED: AF#,AF1,AF2
                    (#1575) *
                    (#1576) *        REGISTERS AFFECTED: BR#,BW1,BW3
                    (#1577) *
A45 04D#E 8A44#C#6  (#1578) VCONT1   LOAD(BR#,3#78(2),L)               ;ADDRESS OF T(1) -1# **
A46 04D1# 8C##4FE9  (#1579) VQUAN1   JUMPS(VQUAN3,AF1)                 ;FIND T(K) > ERROR
```

```
PAGE   41:        MAP MODULES FOR THE  P A R C  ALGORITHM     --- JAN. 3#, 198#

  A47 #4D12 8E##4CEA (#158#)          JUMPS(VQUAN2,AF2)              ;ERROR > ALL T(K)
  A48 #4D14 9###46A8 (#1581)          JUMPC(VQUAN1,AF#)              ;WAIT FOR SIGNAL
  A49 #4D16 922###28 (#1582)          CLEAR(AF#)
  A4A #4D18 948A##9A (#1583)          ADD(BR#,1#,TF)                 ;SEND T(K)
  A4B #4D1A 96##466# (#1584)          JUMP(VQUAN1)
  A4C #4D1C 908A###A (#1585) VQUAN2   ADD(BR#,1#,TF)                 ;SEND OUT(J)
  A4D #4D1E 9A2###2A (#1586)          CLEAR(AF2)
  A4E #4D2# 9C##5168 (#1587)          JUMP(SHIFT)
  A4F #4D22 9E8A###2 (#1588) VQUAN3   ADD(BR#,2,TF)                  ;OUT(J)
  A5# #4D24 A#2###29 (#1589)          CLEAR(AF1)
  A51 #4D26 A28A###2 (#159#) SHIFT    ADD(BR#,2,TF)                  ;EXPN(J)
  A52 #4D28 A4##55E4 (#1591)          JUMPS(VNEGA,G#)                ;
  A53 #4D2A A68A###2 (#1592)          ADD(BR#,2,TF)                  ;B(K)
  A54 #4D2C A###566# (#1593)          JUMP(VNEGA+1)                  ;
  A55 #4D2E AA8A###4 (#1594) VNEGA    ADD(BR#,4,TF)                  ;
  A56 #4D3# AC11##13 (#1595)          MOVB(BW1,BR1)                  ;VHAT(K) ADDRESS
  A57 #4D32 AE91####  (#1596)         ANDB(BW1,BW#,TF)
  A58 #4D34 B131##39 (#1597)          ADDL(BW3,1,TE)                 ;L ADDRESS
                     (#1598) *
                     (#1599) *        ADAPTION
                     (#16##) *
                     (#16#1) *        REGISTERS AFFECTED: BR#,BR1,BW1
                     (#16#2) *
  A59 #4D36 B2C2#426 (#16#3)          LOAD(BR#,SVT79$(1),L,TF)       ;AINV*(1/ALAD-1) **
  A5A #4D38 B48A###2 (#16#4)          ADD(BR#,2,TF)                  ;ALAD
  A5B #4D3A B68A###2 (#16#5)          ADD(BR#,2,TF)                  ;N-2.5
  A5C #4D3C B88A###2 (#16#6)          ADD(BR#,2,TF)                  ;NULL BITS
  A5D #4D3E BA12###1 (#16#7)          SUB(BR1,1)                     ;GET ADDRESS POINTER FOR VHAT BUFFER
  A5E #4D4# BCC4#C## (#16#8)          LOAD(BR#,3#72(2),L,TF)         ;STARTING ADDRESS OF A(I) **
  A5F #4D42 BE99####  (#16#9)         ANDB(BR1,BW#,TF)
  A6# #4D44 C#12###1 (#161#)          SUB(BR1,1)
  A61 #4D46 C28A###2 (#1611)          ADD(BR#,2,TF)
  A62 #4D48 C44E#4## (#1612)          LOAD(BR#,1#24(3),S)
  A63 #4D4A C6##9##2D (#1613)         ADDB(BR#,BR2)
  A64 #4D4C C891##11 (#1614)          MOVB(BW1,BR#,TF)               ;SHAT(K) ADDRESS
  A65 #4D4E CA99####  (#1615)         ANDB(BR1,BW#,TF)               ;VHAT(K-2) ADDRESS
  A66 #4D5# CC12###1 (#1616)          SUB(BR1,1)                     ;
  A67 #4D52 CE44#C## (#1617)          LOAD(BR#,3#72(2),L)            ;**
  A68 #4D54 D#91##11 (#1618)          MOVB(BW1,BR#,TF)               ;A(1) ADDRESS
  A69 #4D56 D291##3A (#1619)          ADDL(BW1,2,TF)                 ;A(2) ADDRESS
  A6A #4D58 D4C4#C#4 (#162#)          LOAD(BR#,3#76(2),L,TF)         ;**
  A6B #4D5A D699####  (#1621)         ANDB(BR1,BW#,TF)
  A6C #4D5C D891##3A (#1622)          ADDL(BW1,2,TF)
  A6D #4D5E DA12###1 (#1623)          SUB(BR1,1)
```

```
A6E 04D60 DC8A0002 (01624)          ADD(BR0,2,TF)
A6F 04D62 DE990000 (01625)          ANDB(BR1,BW0,TF)
A70 04D64 E091003A (01626)          ADDL(BW1,2,TF)
A71 04D66 E21A0005 (01627)          ADD(BR1,5)
                   (01628) *
                   (01629) *        END OF BUFFER ?
                   (01630) *
                   (01631) *        FLAGS SENSED: G1,AF0,AF3
                   (01632) *
A72 04D68 E4001EE8 (01633) VCHECK   JUMPS(VLOOP,AF0)                ;JUMP IF AF0 SET --- PARC LOOP NOT END
A73 04D6A E6007EEB (01634)          JUMPS(VEND,AF3)                 ;JUMP IF BITCOUNT < 0
A74 04D6C E800E5A5 (01635)          JUMPC(VCHECK,G1)                ;WAIT FOR SIGNAL
                   (01636) *
                   (01637) *        END
                   (01638) *
                   (01639) *        REGISTERS AFFECTED: BR0,BW1,BW3
                   (01640) *
A75 04D6E EB310039 (01641) VEND     ADDL(BW3,1,TE)                  ;OUTPUT END CODE
A76 04D70 ECC20042E (01642)         LOAD(BR0,SVT03S(1),L,TF)        ;1./(2.**15)
A77 04D72 EE520404 (01643)          LOAD(BR1,SVT63S(1),L)           ;
A78 04D74 F0910013 (01644)          MOVB(BW1,BR1,TF)                ;BITCOUNT
A79 04D76 F291003C (01645)          ADDL(BW1,4,TF)                  ;RMS
A7A 04D78 F491003A (01646)          ADDL(BW1,2,TF)                  ;STATE VARIABLE
A7B 04D7A F6A00002 (01647)          ADD(BR0,2)                      ;
A7C 04D7C F9110011 (01648)          MOVB(BW1,BR0,TE)                ;# OF SAMPLES
A7D 04D7E FA200031 (01649)          CLEAR(RI)
A7E 04D80 FC000020 (01650)          NOP
                   (01651) *
           0000C92 (01652) VPCTXSA=0C                               ;CHAIN ANCHOR
                   (01653) *
    04D82          (01654)          END
                   (01655) *
                   (01656)          EVEN
                   (01657) *
                   (01658) *        STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                   (01659) *
    04D82 00000000 (01660) VPCTXSI DATA     1F'0.0'
                   (01661) *
           0000100 (01662) VPCTXSZ=0L-VPCTXS                        ;COMPUTE MODULE SIZE
                   (01663) *
           0004D84 (01664)          CON3S=0L
                   (01665) *
                   (01666)          EJECT
```

```
(01667) *
(01668) *       ******************************************************
(01669) *       *                                                    *
(01670) *       *      PITCH EXTRACTION ADAPTIVE RESIDUAL CODER      *
(01671) *       *                    P A R C                         *
(01672) *       *                    RECEIVER                        *
(01673) *       *                                                    *
(01674) *       ******************************************************
(01675) *
(01676) *       *  DESCRIPTION:
(01677) *
(01678) *       THIS IS A SPEECH CODING PROGRAM IN MAP-300 ARITHMETIC PROCESSOR
(01679) *       FOR DIGITAL TRANSMISSION OF SPEECH AT A RATE OF 9600 BITS PER SECOND.
(01680) *       THE ALGORITHM COMBINES PITCH EXTRACTION LOOP, PITCH COMPENSATING
(01681) *       ADAPTIVE QUANTIZER, SEQUENTIALLY ADAPTIVE PREDICTOR.
(01682) *
(01683) *       1. THE SEQUANTIALLY ADAPTIVE PREDICTOR USING BACKWARD ADAPTION
(01684) *          IS USED TO FORM AN ESTIMATE OF PITCH REDUCED SIGNAL.
(01685) *       2. THE ERROR OF THIS ESTIMATE IS RECONSTRUCTED BY INVERSE QUANTIZER.
(01686) *       3. THE REDUNDANCE OF THE CURRENT SAMPLE IS CALCULATED BY PITCH
(01687) *          EXTRACTION LOOP.
(01688) *       4. THE RECONSTRUCTED SPEECH IS THE SUM OF THIS ERROR, THE ESTIMATE
(01689) *          AND THE REDUNDANCE.
(01690) *
(01691) *       * SYMBOLIC ABBREVIATIONS:
(01692) *
(01693) *         SH$RC --- THE LOCATION OF SHAT POINTER
(01694) *         T$RC  --- THE LOCATION OF PITCH PERIOD FOR RECEIVER
(01695) *         VH$RC --- THE LOCATION OF VHAT POINTER
(01696) *         IN$RC --- THE LOCATION OF (PITCH REPETITION SIZE - 1)
(01697) *
(01698) *       * BUFFERS:
(01699) *
(01700) *       1. Q BUFFER: IT CONTAINS INPUT QUANTIZING LEVELS.
(01701) *          --- IN 16-BIT INTEGER*2 FORMAT --- DOUBLE BUFFERS ---
(01702) *       2. VHAT BUFFER: IT CONTAINS RECONSTRUCTED REDUCED SPEECH.
(01703) *          --- CIRCULAR BUFFER --- IN SHORT-REAL FORMAT ---
(01704) *       3. SHAT BUFFER: IT CONTAINS RECONSTRUCTED SPEECH.
(01705) *          --- CIRCULAR BUFFER --- IN SHORT-REAL FORMAT ---
(01706) *       4. AOM BUFFER: IT CONTAINS UPSAMPLED SHAT DATA.
(01707) *          --- DOUBLE BUFFER --- SHORT REAL FORMAT ---
(01708) *       5. PARAMETER BUFFER: IT CONTAINS THE PARAMETERS FOR INVERSE QUANTIZER.
(01709) *          SEQUENCE : A(8),(ENPN(J),OUT(J)) --- IN LONG-REAL FORMAT ---
(01710) *
```

```
                    (01711) *       * RESTRICTION:
                    (01712) *
                    (01713) *       1. THE ORDER OF THE PREDICTOR IS 4.
                    (01714) *       2. INITIALLY, THE SHAT BUFFER SHOULD BE IN FULL CONDITION.
                    (01715) *       3. THE LOCATIONS OF SOME ARRAYS ARE AS FOLLOWS:
                    (01716) *          SHAT BUFFER --- 2048(2) ... 3071(2)
                    (01717) *          VHAT BUFFER --- 2048(3) ... 3071(3)
                    (01718) *          AOM BUFFER --- 3584(2)-4095(2)
                    (01719) *          PARAMETER BUFFER --- 4096(2) ...
                    (01720) *          WHERE QUANTIZER LEVELS ARE DEFINED AS FOLLOWS:
                    (01721) *          FROM LOWEST ..... TO ..... HIGHEST
                    (01722) *          Q(N),Q(N-1),...,Q(N/2+2),Q(1),Q(2),....,Q(N/2+1)
                    (01723) *
                    (01724) *       * CALL TO THIS FUNCTION:
                    (01725) *
                    (01726) *         MAP = PCRC(SA,U,V)
                    (01727) *         WHERE  SA --- SCALAR LOCATION OF ENCD. BETA
                    (01728) *                U  --- BUFFER # FOR INPUT Q BUFFER
                    (01729) *                V  --- AOM BUFFER #
                    (01730) *
                    (01731) *
                    (01732) *       DISPATCH TABLE ENTRY
                    (01733) *
          000000F3  (01734)         FCB243=243                      ;FCB=243
          00009B9A  (01735)         #L=AFDT$+3*2*(FCB243-128)       ;STARTING LOCATION OF AFDT FOR 243
                    (01736) *
    009B9A 001E4D86 (01737)         ADDR      PCRC$(R7,1)           ;APU=PCRC
    009B9C 001E4EC2 (01738)         ADDR      VPCRC$(R7,1)          ;APS=VPCRC
    009B9E 001021FC (01739)         ADDR      CSPU$NOS(,1,0)        ;NO CSPU SUPPORT
                    (01740) *
                    (01741) *       APU & APS MODULE
                    (01742) *
          00004D84  (01743)         #L=CON3$
                    (01744) *
                    (01745) *       APU MODULE
                    (01746) *
                    (01747)         EVEN                            ;START ON WORD BOUNDARY
    04D84 0000      (01748)         DATA      PCRC$SA               ;STARTING ADDRESS
    04D85 009A      (01749)         DATA      PCRC$SZ               ;SIZE
                    (01750) *
                    (01751) PCRC$   BEGIN     APU(PCRC)             ;START OF APU MODULE
          00000003  (01752)         #M=3
          00000000  (01753)         #A=0
                    (01754) *
```

```
                    (01755) *       DATA TRANSFER FROM SHAT BUFFER TO AOM BUFFER
                    (01756) *
                    (01757) *       REGISTERS USED:
                    (01758) *       ADM1: A0,A1,A2,A5,A6,A7,M1,M7,EXO
                    (01759) *       ADM2: A0,A1,A7,M1,M7,EXO
                    (01760) *
                    (01761) *       INPUT SEQUENCE: 2047/16**3,-2048/16**3,TRANSFER SIZE/2-0.5,SHAT(K-1),SHA
                    (01762) *       OUTPUT SEQUENCE:  (SHAT(K-1)+SHAT(K))/2,SHAT(K)
                    (01763) *
A00 04D86 16601660  (01764) PCRCSSA K(0.5)
A01 04D88 00F50000  (01765)         MOV(IQA,A5) \ NOP               ;A5=2047/(16**3)
A02 04D8A 00F60000  (01766)         MOV(IQA,A6) \ NOP               ;A6=-2048/(16**3)
A03 04D8C 16891689  (01767)         MOV(M1),K(1)
A04 04D8E 00F20000  (01768)         MOV(IQA,A2) \ NOP               ;A2=TRANSFER SIZE/2-0.5
A05 04D90 00F00000  (01769)         MOV(IQA,A0) \ NOP               ;SHAT(K-1)
A06 04D92 6D170897  (01770)         MOV(A7),MIN(A0,A5) \ MOV(R,A7) ;LIMITING
A07 04D94 66100000  (01771)         MOV(A0),MAX(A0,A6) \ NOP        ;LIMITING
A08 04D96 00F10000  (01772) TRAN    MOV(IQA,A1) \ NOP
A09 04D98 6D200000  (01773)         MIN(A1,A5) \ NOP
A0A 04D9A 66310000  (01774)         MOV(A1),MAX(A1,A6) \ NOP
A0B 04D9C 00980000  (01775)         MOV(R,EXO) \ NOP
A0C 04D9E 41110000  (01776)         MOV(A1),ADD(A0,A1) \ NOP
A0D 04DA0 00F00858  (01777)         MOV(R,M7) \ MOV(EXI,EXO)
A0E 04DA2 04E00000  (01778)         MUL(M1,M7) \ NOP
A0F 04DA4 4F400000  (01779)         SUB(A2,A7) \ NOP
A10 04DA6 00500000  (01780)         MOV(EXI,A0) \ NOP
A11 04DA8 00BC0000  (01781)         MOV(P,0Q) \ NOP
A12 04DAA 00920850  (01782)         MOV(R,A2) \ MOV(EXI,0Q)
A13 04DAC 901E0000  (01783)         JUMPC(TRAN,T1)                  ;LOOPING
                    (01784) *
                    (01785) *       INITIALIZATION
                    (01786) *
                    (01787) *       REGISTERS AFFECTED:
                    (01788) *
                    (01789) *       ADM1: A0,A1,A2,M0,M1,M3,M5,M6,M7
                    (01790) *       ADM2: A0,A2,A3,A4,A6,A7,M0,M2,M6
                    (01791) *
                    (01792) *       INPUT SEQUENCE: ENCD.BETA,RMS,STATE VARIABLE,UB*2**15/HL,RMSMIN,N-2.5
                    (01793) *                       ALAD,SMIN,UB,G,1-ALP,ALP,AINV*(1/ALAD-1),2**14,PITCH REP
                    (01794) *
                    (01795) *                                       ADM1                ADM2
                    (01796) *                                       ***************     ****************
                    (01797) *
A14 04DAE 00F10000  (01798)         MOV(IQA,A1) \ NOP               ;A1=ENCD. BETA
```

```
                    (01711) *       * RESTRICTION:
                    (01712) *
                    (01713) *       1. THE ORDER OF THE PREDICTOR IS 4.
                    (01714) *       2. INITIALLY, THE SHAT BUFFER SHOULD BE IN FULL CONDITION.
                    (01715) *       3. THE LOCATIONS OF SOME ARRAYS ARE AS FOLLOWS:
                    (01716) *          SHAT BUFFER --- 2048(2) ... 3071(2)
                    (01717) *          VHAT BUFFER --- 2048(3) ... 3071(3)
                    (01718) *          AOM BUFFER --- 3584(2)-4095(2)
                    (01719) *          PARAMETER BUFFER --- 4096(2) ...
                    (01720) *          WHERE QUANTIZER LEVELS ARE DEFINED AS FOLLOWS:
                    (01721) *          FROM LOWEST ..... TO ..... HIGHEST
                    (01722) *          Q(N),Q(N-1),...,Q(N/2+2),Q(1),Q(2),....,Q(N/2+1)
                    (01723) *
                    (01724) *       * CALL TO THIS FUNCTION:
                    (01725) *
                    (01726) *         MAP = PCRC(SA,U,V)
                    (01727) *         WHERE  SA --- SCALAR LOCATION OF ENCD. BETA
                    (01728) *                U  --- BUFFER # FOR INPUT Q BUFFER
                    (01729) *                V  --- AOM BUFFER #
                    (01730) *
                    (01731) *
                    (01732) *       DISPATCH TABLE ENTRY
                    (01733) *
          ######F3  (01734)         FCB243=243                      ;FCB=243
          #####89A  (01735)         #L=AFDT$+3*2*(FCB243-128)       ;STARTING LOCATION OF AFDT FOR 243
                    (01736) *
    ##89A ##1E4D86  (01737)         ADDR      PCRC$(R7,1)           ;APU=PCRC
    ##89C ##1E4EC2  (01738)         ADDR      VPCRC$(R7,1)          ;APS=VPCRC
    ##89E ##1021FC  (01739)         ADDR      CSPU$NOS(,1,0)        ;NO CSPU SUPPORT
                    (01740) *
                    (01741) *       APU & APS MODULE
                    (01742) *
          ####4D84  (01743)         #L=CON3$
                    (01744) *
                    (01745) *       APU MODULE
                    (01746) *
                    (01747)         EVEN                            ;START ON WORD BOUNDARY
    04D84 ####      (01748)         DATA      PCRC$SA               ;STARTING ADDRESS
    04D85 ##9A      (01749)         DATA      PCRC$SZ               ;SIZE
                    (01750) *
                    (01751) PCRC$   BEGIN     APU(PCRC)             ;START OF APU MODULE
          ########  (01752)         #M=3
          ########  (01753)         #A=0
                    (01754) *
```

```
A15 04DB0 322000F2 (01799)          NORM(A1) \ MOV(IQA,A2)          ;                       A2=RMS
A16 04DB2 08E90000 (01800)          MOV(IQA,M1) \ NOP               ;M1=STATE VARIABLE
A17 04DB4 08EF0000 (01801)          MOV(IQA,M7) \ NOP               ;M7=UB*2**15/HL
A18 04DB6 088B00F6 (01802)          MOV(R,M3) \ MOV(IQA,A6)         ;M3=ENCD. BETA          A6=RMSMIN
A19 04DB8 85E000F0 (01803)          MUL(M3,M7) \ MOV(IQA,A0)        ;                       A0=N-2.5
A1A 04DBA 08ED1680 (01804)          MOV(IQA,M5) \ K(1)              ;M5=ALAD
A1B 04DBC 08EE0814 (01805)          MOV(IQA,M6) \ MOV(ZERO,A4)      ;M6=SMIN                A4=SAMPLE COUNT
A1C 04DBE 08F20000 (01806)          MOV(IQA,A2) \ NOP               ;A2=UB
A1D 04DC0 08E80000 (01807)          MOV(IQA,M0) \ NOP               ;M0=G
A1E 04DC2 08B108E8 (01808)          MOV(P,A1) \ MOV(IQA,M0)         ;A1=UB+BETA             M0=1-ALP
A1F 04DC4 4A2008EE (01809)          SUB(A1,A2) \ MOV(IQA,M6)        ;                       M6=ALP
A20 04DC6 08F00897 (01810)          MOV(IQA,A0) \ MOV(R,A7)         ;A0=AINV(1/ALAD-1)      A7=1
A21 04DC8 00000EA (01811)           NOP \ MOV(IQA,M2)               ;                       M2=2**14
A22 04DCA 088B00F3 (01812)          MOV(R,M3) \ MOV(IQA,A3)         ;M3=BETA                A3=PITCH REPETITION
                    (01813) *
                    (01814) *       CHECK PITCH REPETITION MODE
                    (01815) *
                    (01816) *       FLAGS AFFECTED: G3,AF1
                    (01817) *
                    (01818) *       REGISTERS AFFECTED:
                    (01819) *
                    (01820) *       ADM1: A5
                    (01821) *       ADM2: A3
                    (01822) *
                    (01823) *       INPUT SEQUENCE: PITCH REPETITION CODE,INPUT PITCH REPETITION DATA
                    (01824) *       OUTPUT SEQUENCE: OUTPUT PITCH REPETITION DATA
                    (01825) *
A23 04DCC 08F50000 (01826)          MOV(IQA,A5) \ NOP               ;READ PITCH REPETITION CODE
A24 04DCE 02A00000 (01827)          R(A5) \ NOP                     ;CHECK PITCH RREPETITION
A25 04DD0 08800000 (01828)          MOV(R,NULL) \ NOP               ;
A26 04DD2 901E0098 (01829)          JUMPC(SIGNAL,T1)                ;JUMP IF NO PITCH REPETITION
A27 04DD4 20472047 (01830)          SET(G3)                         ;SIGNAL APS OF PITCH REPETITION MODE
A28 04DD6 08FC4F60 (01831) GAP      MOV(IQA,0Q) \ SUB(A3,A7)        ;MOVING A PITCH REPETITION-BLOCK OF SHAT
A29 04DD8 00000893 (01832)          NOP \ MOV(R,A3)                 ;DECREMENT COUNTER
A2A 04DDA 901F0028 (01833)          JUMPC(GAP,T2)                   ;LOOPING
A2B 04DDC 10000085 (01834)          JUMP(CKEND)                     ;JUMP TO READ L
                    (01835) *
                    (01836) *       PREDICTOR
                    (01837) *
                    (01838) *       TO PREDICT THE VALUE OF CURRENT SAMPLE
                    (01839) *
                    (01840) *       REGISTERS AFFECTED:
                    (01841) *
                    (01842) *       ADM1: A1,A5,M2,M7
```

```
                    (01843) *       ADM2: A1,A2,A3,M1,M7,EXO
                    (01844) *
                    (01845) *       FLAGS AFFECTED: AF0
                    (01846) *
                    (01847) *       INPUT SEQUENCE: A(I),VHAT(K-I)
                    (01848) *
A2C 04DDE 20402048  (01849) PRE     SET(AF0)                        ;SIGNAL APS TO PROCESS PARC-RECEIVER
A2D 04DE0 08EA020D  (01850)         MOV(IQA,M2) \ R(A0)             ;M2=A(I)                R=N-2.5
A2E 04DE2 08CF0000  (01851)         MOV(IQ,M7) \ NOP                ;M7=VHAT(K-1)           ------
A2F 04DE4 856000F1  (01852)         MUL(M2,M7) \ MOV(IQA,A1)        ;P1=A(I)*VHAT(K-1)      A1=VHAT(K-1)
A30 04DE6 08151233  (01853)         MOV(ZERO,A5) \ MOV(A3),ABS(A1) ;A5=0.                   A3=N-2.5,R=ABS(A1)
A31 04DE8 0000008F  (01854)         NOP \ MOV(R,M7)                 ;------                 M7=ABS(VHAT(K-1))
A32 04DEA 08B1846D  (01855)         MOV(P,A1) \ MUL(M0,M7)          ;A1=P1                  P2=(1-ALP)*M7
A33 04DEC 41A04E40  (01856)         ADD(A5,A1) \ SUB(A2,A6)         ;R1=PRE+P1              R2=RMS-RMSMIN
A34 04DEE 08EA0000  (01857)         MOV(IQA,M2) \ NOP               ;M2=A(2)                ------
A35 04DF0 08EF0000  (01858)         MOV(IQA,M7) \ NOP               ;M7=VHAT(K-2)           ------
A36 04DF2 85600089  (01859)         MUL(M2,M7) \ MOV(R,M1)          ;P3=A(2)*VHAT(K-2)      M1=R2   INCRE ?
A37 04DF4 0000D4D1  (01860)         NOP \ MOV(A1),MUL(M1,M6)        ;------                 A1=P2   P4=R2*ALP
A38 04DF6 0000462D  (01861)         NOP \ ADD(A1,A6)                ;------                 R3=P2+RMSMIN
A39 04DF8 08B10B2   (01862)         MOV(P,A1) \ MOV(P,A2)           ;A1=P3                  A2=P4
A3A 04DFA 41B54231  (01863)         MOV(A5),ADD(A5,A1) \ MOV(A1),ADD(A1,A2) ;EXO=NQ-0.5  A1=R3
A3B 04DFC 0895008F  (01864)         MOV(R,A5) \ MOV(R,M7)           ;A5=NEW PRE             M7=NEW RMS
A3C 04DFE 00000089  (01865)         NOP \ MOV(R,M1)                 ;------                 M1=NEW RMS
A3D 04E00 00000892  (01866)         NOP \ MOV(R,A2)                 ;------                 A2=NEW RMS
A3E 04E02 00000898  (01867)         NOP \ MOV(R,EXO)                ;------                 EXO=NEW RMS
                    (01868) *
                    (01869) *       PREDICTOR LOOP
                    (01870) *
A3F 04E04 08EA04E0  (01871) PRE1    MOV(IQA,M2) \ MUL(M1,M7)        ;M2=A(I)                P=RMS**2
A40 04E06 08EF0000  (01872)         MOV(IQA,M7) \ NOP               ;M7=VHAT(K-I)           -----
A41 04E08 85600000  (01873)         MUL(M2,M7) \ NOP                ;P=A(I)*VHAT(K-I)       ------
A42 04E0A 08B10081  (01874)         MOV(P,A1) \ MOV(P,A1)           ;A1=A(I)*VHAT(K-I)      A1=RMS**2
A43 04E0C 41A04F60  (01875)         ADD(A5,A1) \ SUB(A3,A7)         ;R=NEW PRE              -----
A44 04E0E 08950893  (01876)         MOV(R,A5) \ MOV(R,A3)           ;A5=RENEW PRE           A3=COUNTER-1
A45 04E10 901F00F3  (01877)         JUMPC(PRE1,T2)                  ;NED7
                    (01878) *
                    (01879) *       INVERSE QUANTIZER
                    (01880) *
                    (01881) *       REGISTERS AFFECTED:
                    (01882) *
                    (01883) *       ADM1: A1,A2,A3,A4,A5,M1,M2,M4,M7,EXO
                    (01884) *       ADM2: A1,A3,A4,A5,M1,M4,M5,M7
                    (01885) *
                    (01886) *       FLAGS AFFECTED: AF0,AF1
```

```
                    (0187) *        FLAGS SENSED: AF0
                    (0188) *
                    (0189) *        INPUT SEQUENCE: EXPN(J),OUT(J),SHAT(K-T)
                    (018A) *        OUTPUT SEQUENCE: VHAT(K),SHAT(K)
                    (018B) *
A46 04E12 00000000  (018C) QUAN     NOP
A47 04E14 910000A6  (018D)          JUMPS(QUAN,AF0)
A48 04E16 20402040  (018E)          SET(AF0)                        ;SIGNAL APS TO GO TO NEXT OUT(J)
A49 04E18 004A4FA0  (018F)          MOV(EXI,M2) \ SUB(A5,A7)
A4A 04E1A 00000095  (0190)          NOP \ MOV(R,A5)
A4B 04E1C 981F0046  (0191)          JUMPC(QUAN,T2)
A4C 04E1E 20402049  (0192)          SET(AF1)                        ;SIGNAL APS TO OUTPUT OUT(J) & EXPN(J)
A4D 04E20 054000AF  (0193)          MUL(M2,M6) \ MOV(P,M7)          ;P=RMS*SMIN             M7=RMS**2
A4E 04E22 00EF2E20  (0194)          MOV(IQA,M7) \ RCP(A1)           ;M7=EXPN(J)             R=1/RMS**2+DEL
A4F 04E24 00000089  (0195)          NOP \ MOV(R,M1)                 ;-----                  M1=F0
A50 04E26 000004E0  (0196)          NOP \ MUL(M1,M7)                ;-----                  P2=F0*RMS**2
A51 04E28 04F116A0  (0197)          MOV(A1),MUL(M1,M7) \ K(2)       ;A1=RMS*SMIN            R=2
A52 04E2A 00EC00B3  (0198)          MOV(IQA,M4) \ MOV(P,A3)         ;M4=OUT(J)              A3=P2
A53 04E2C 00004831  (0199)          NOP \ MOV(A1),SUB(A1,A3)        ;-----                  A1=2
A54 04E2E 0000008D  (019A)          NOP \ MOV(R,M5)                 ;-----                  M5=2-P2
A55 04E30 049384A0  (019B)          MOV(A3),MUL(M1,M4) \ MUL(M1,M5) ;A3=SIZE*EXPN(J)        P=F0*(2-P2)
A56 04E32 63200020  (019C)          MAX(A1,A3) \ NOP
A57 04E34 00EF0000  (019D)          MOV(IQA,M7) \ NOP               ;SHAT(K-T)
A58 04E36 00940000  (019E)          MOV(R,A4) \ NOP
A59 04E38 00B200A9  (019F)          MOV(P,A2) \ MOV(P,M1)           ;A2=EQ                  M1=F1
A5A 04E3A 05EC0000  (01A0)          MOV(M4),MUL(M3,M7) \ NOP
A5B 04E3C 454984E0  (01A1)          MOV(M1),ADD(A2,A5) \ MUL(M1,M7) ;M1=EQ                  P=F1*RMS**2
A5C 04E3E 000000B3  (01A2)          NOP \ MOV(P,A3)                 ;-----                  A3=F1*RMS**2
A5D 04E40 00004B20  (01A3)          NOP \ SUB(A1,A3)                ;-----                  R=2-A3
A5E 04E42 0000008D  (01A4)          NOP \ MOV(R,M5)                 ;-----                  M5=R
A5F 04E44 009C84A0  (01A5)          MOV(R,OQ) \ MUL(M1,M5)          ;OUTPUT VHAT(K)         P=F2
A60 04E46 00B30000  (01A6)          MOV(P,A3) \ NOP
A61 04E48 433100A9  (01A7)          MOV(A1),ADD(A1,A3) \ MOV(P,M1)  ;A1=EQ                  M1=F2
A62 04E4A 009C0000  (01A8)          MOV(R,OQ) \ NOP                 ;SHAT(K)
A63 04E4C 000004E0  (01A9)          NOP \ MUL(M1,M7)                ;EXO=G*EQ               P=F2*RMS**2
A64 04E4E 000000B3  (01AA)          NOP \ MOV(P,A3)                 ;-----                  A3=P
A65 04E50 00004B20  (01AB)          NOP \ SUB(A1,A3)                ;-----                  R=2-P
A66 04E52 0000008D  (01AC)          NOP \ MOV(R,M5)
A67 04E54 000084A0  (01AD)          NOP \ MUL(M1,M5)
A68 04E56 000000A9  (01AE)          NOP \ MOV(P,M1)                 ;-----                  M1=F3
A69 04E58 000000B8  (01AF)          NOP \ MOV(P,EXO)
A6A 04E5A 004F4780  (01B0)          MOV(EXI,M7) \ ADD(A4,A7)
A6B 04E5C 84600000  (01B1)          MUL(M0,M7) \ NOP
A6C 04E5E 00AA2000  (01B2)          MOV(P,M2) \ NOP
```

```
A6D 04E60 85000080C (01931)          MUL(M2,M4) \ MOV(R,M4)
A6E 04E62 00800894 (01932)          MOV(P,EXO) \ MOV(R,A4)
                   (01933) *
                   (01934) *        ADAPTATION
                   (01935) *
                   (01936) *        REGISTERS AFFECTED:
                   (01937) *
                   (01938) *        ADM1: A1,A3,A5,M2
                   (01939) *        ADM2: A3,M1,M5,M7,EXO
                   (01940) *
                   (01941) *        INPUT SEQUENCE: A(I), VHAT(K-I)
                   (01942) *        OUTPUT SEQUENCE: A(I)
                   (01943) *
A6F 04E64 00F50849 (01944)          MOV(IQA,A5) \ MOV(EXI,M1)      ;A5=A(I)                P1=G*EQ/RMS**2
A70 04E66 40A00000 (01945)          ADD(A5,A0) \ NOP               ;R1=A(1)+ANIV*0.00787 ------
A71 04E68 008A0000 (01946)          MOV(R,M2) \ NOP                ;M2=R1                  M1=P1
A72 04E6A 852008EF (01947)          MUL(M2,M5) \ MOV(IQA,M7)       ;P2=ALAD*R1             M7=VHAT(K-1)
A73 04E6C 00810000 (01948)          MOV(P,A1) \ NOP
A74 04E6E 00EA84E0 (01949)          MOV(IQA,M2) \ MUL(M1,M7)       ;M1=A(2)                P3=VHAT(K-1)*P1
A75 04E70 85200088 (01950)          MUL(M2,M5) \ MOV(P,EXO)        ;A1=P2                  EXO=P3
A76 04E72 00530ED (01951)           MOV(EXI,A3) \ MOV(IQA,M5)      ;A3=VHAT(K-1)*ERR       M5=VHAT(K-2)
A77 04E74 41600400 (01952)          ADD(A3,A1) \ MUL(M1,M5)        ;R=NEW A(1)             P=VHAT(K-2)*ERR
A78 04E76 009C0000 (01953)          MOV(R,OQ) \ NOP                ;OUTPUT A(1)            -----
A79 04E78 00810088 (01954)          MOV(P,A1) \ MOV(P,EXO)         ;A1=ALAD*A(2)           EXO=VHAT(K-2)*ERR
A7A 04E7A 00530200 (01955)          MOV(EXI,A3) \ R(A0)            ;A3=VHAT(K-2)*ERR       R=N-2.5
A7B 04E7C 41600000 (01956)          ADD(A3,A1) \ NOP               ;R=NEW A(2)
A7C 04E7E 009C0093 (01957)          MOV(R,OQ) \ MOV(R,A3)          ;OUTPUT A(2)            A3=N-2.5
A7D 04E80 00EA0000 (01958) ADAP     MOV(IQA,M2) \ NOP              ;M1=A(I)
A7E 04E82 852008EF (01959)          MUL(M2,M5) \ MOV(IQA,M7)       ;P=A(I)*ALAD            M7=VHAT(K-I)
A7F 04E84 00000E0 (01960)           NOP \ MUL(M1,M7)               ;-----                  P=VHAT(K-I)*ERR
A80 04E86 00810088 (01961)          MOV(P,A1) \ MOV(P,EXO)         ;A1=A(I)*ALAD           EXO=VHAT(K-I)*ERR
A81 04E88 00534F60 (01962)          MOV(EXI,A3) \ SUB(A3,A7)       ;A3=VHAT(K-I)*ALAD      COUNTER=COUNTER-1
A82 04E8A 41600000 (01963)          ADD(A3,A1) \ NOP               ;R=NEW A(I)
A83 04E8C 009C0093 (01964)          MOV(R,OQ) \ MOV(R,A3)          ;OUTPUT NEW A(I)
A84 04E8E 901F007D (01965)          JUMPC(ADAP,T2)                 ;END ?
                   (01966) *
                   (01967) *        CHECK END OF RECEIVER
                   (01968) *
                   (01969) *        REGISTERS AFFECTED:
                   (01970) *
                   (01971) *        ADM1:
                   (01972) *        ADM2: A5
                   (01973) *
A85 04E90 000000F5 (01974) CKEND    NOP \ MOV(IQA,A5)              ;                       A5=L
```

```
PAGE   5Ø:        MAP MODULES FOR THE  P A R C  ALGORITHM     --- JAN. 3Ø, 198Ø

  A86 Ø4E92 ØØØØ32AØ (Ø1975)         NOP \ NORM(A5)                ;
  A87 Ø4E94 ØØØØØ8ØF (Ø1976)         NOP \ MOV(R,M7)               ;                       M7=L
  A88 Ø4E96 ØØØØØ56Ø (Ø1977)         NOP \ MUL(M2,M7)              ;
  A89 Ø4E98 ØØØØØ2B5 (Ø1978)         NOP \ MOV(A5),R(A5)           ;                       CHECK END CODE
  A8A Ø4E9A ØØØØØ88Ø (Ø1979)         NOP \ MOV(R,NULL)             ;
  A8B Ø4E9C ØØØØØ8B5 (Ø198Ø)         NOP \ MOV(P,A5)               ;                       A5=L
  A8C Ø4E9E 9Ø1FØØ2C (Ø1981)         JUMPC(PRE,T2)                 ;JUMP IF NOT END CODE
                     (Ø1982) *
                     (Ø1983) *       END OF RECEIVER
                     (Ø1984) *
                     (Ø1985) *       FLAGS AFFECTED: AF3,RA
                     (Ø1986) *
                     (Ø1987) *       REGISTERS AFFECTED:
                     (Ø1988) *
                     (Ø1989) *       ADM1:
                     (Ø199Ø) *       ADM2: A2,M3
                     (Ø1991) *
                     (Ø1992) *       INPUT SEQUENCE: 1/2**15
                     (Ø1993) *       OUTPUT SEQUENCE: STATE VARIABLE,RMS,# OF SAMPLES
                     (Ø1994) *
  A8D Ø4EAØ 2Ø4B2Ø4B (Ø1995)         SET(AF3)                      ;SIGNAL APS END OF RECEIVER
  A8E Ø4EA2 ØØØØØ8EB (Ø1996)         NOP \ MOV(IQA,M3)             ;                       M3=1./(2.**15)
  A8F Ø4EA4 ØØØØ858Ø (Ø1997)         NOP \ MUL(M3,M4)              ;
  A9Ø Ø4EA6 Ø28ØØ24Ø (Ø1998)         R(A4) \ R(A2)                 ;
  A91 Ø4EA8 Ø89CØ89C (Ø1999)         MOV(R,OQ)                     ;STATE VARIABLE         RMS
  A92 Ø4EAA ØØØØ9ØB2 (Ø2ØØØ)         NOP \ MOV(P,A2)               ;
  A93 Ø4EAC ØØØØ3A4Ø (Ø2ØØ1)         NOP \ ALIGN(A2)               ;
  A94 Ø4EAE ØØØØØ89C (Ø2ØØ2)         NOP \ MOV(R,OQ)               ;                       OUTPUT # OF SAMPLES
  A95 Ø4EBØ 2Ø322Ø32 (Ø2ØØ3)         CLEAR(RA)                     ;CONTINUE
  A96 Ø4EB2 ØØØØØØØØ (Ø2ØØ4)         NOP
  A97 Ø4EB4 1ØØØØØØØ (Ø2ØØ5)         JUMP(Ø)
                     (Ø2ØØ6) *
  A98 Ø4EB6 2Ø492Ø49 (Ø2ØØ7) SIGNAL  SET(AF1)                       ;SIGNAL APS NO PITCH REPETITION
  A99 Ø4EB8 1ØØØØØ85 (Ø2ØØ8)         JUMP(CKEND)                   ;
                     (Ø2ØØ9) *
            ØØØØØØ9A (Ø2Ø1Ø)         PCRC$SZ=#A-PCRC$SA            ;COMPUTE THE SIZE OF THIS MODULE
      Ø4EBA          (Ø2Ø11)         END
                     (Ø2Ø12) *
                     (Ø2Ø13)         EJECT
```

```
                    (02014) *
                    (02015) *       APS MODULE OF PCRC PROGRAM
                    (02016) *
                    (02017)         EVEN                            ;START ON WORD BOUNDARY
     04EBA 00004FBE (02018)         ADDR    VPCRC$I                 ;PTR TO CONSTR INSTR BLOCK
     04EBC 00004EE8 (02019)         ADDR    VPCRC$+2*VPCRC$S        ;STARTING ADDR. OF SCALARS
     04EBE 0001     (02020)         DATA    1                       ;ONE SCALAR
     04EBF 0100     (02021)         DATA    VPCRC$Z                 ;SIZE
     04EC0 00004F06 (02022)         ADDR    VPCRC$A                 ;CHAIN ANCHOR
                    (02023)         EVEN
                    (02024) *
                    (02025) *
                    (02026) VPCRC$  BEGIN   APS(VPCRC)
                    (02027) *
  A00 04EC2 00300032 (02028)        SET(RA)
                    (02029) *
                    (02030) *       UP SAMPLING
                    (02031) *
                    (02032) *       REGISTERS AFFECTED: BR0,BR1,BR2,BR3,BW0,BW1,BW2
                    (02033) *
  A01 04EC4 0240003F (02034)        LOAD(BR0,1023)                  ;**
  A02 04EC6 04010011 (02035)        MOVB(BW0,BR0)                   ;MASK
  A03 04EC8 06400000 (02036)        LOAD(BR0,2048)                  ;**
  A04 04ECA 00110011 (02037)        MOVB(BW1,BR0)                   ;STARTING LOCATION OF RECEIVER BUFFER
  A05 04ECC 0AC20468 (02038)        LOAD(BR0,SVT115$(1),L,TF)       ;2047/(16**3)
  A06 04ECE 0C0A0002 (02039)        ADD(BR0,2,TF)                   ;-2048/(16**3)
  A07 04ED0 0EC20446 (02040)        LOAD(BR0,SVT98$(1),L,TF)        ;TRANSFERR SIZE/2-0.5
      00004ED3 (02041)              DASRC=0L+1
  A08 04ED2 10DC0782 (02042)        LOAD(BR1,1922(2),S,TF)          ;AOM POINTER (+S) DA(K-1)
  A09 04ED4 12720000 (02043)        LOAD(BR3,[2])                   ;AOM BUFFER
  A0A 04ED6 14600000 (02044)        LOAD(BR2,M$$)                   ;SIZE - 1
  A0B 04ED8 16320000 (02045)        SUB(BR3,M$$)                    ;SHAT(K-1)
  A0C 04EDA 18210017 (02046)        MOVB(BW2,BR3)                   ;
  A0D 04EDC 1A1A0001 (02047) VTRAN  ADD(BR1,1)
  A0E 04EDE 1C190000 (02048)        ANDB(BR1,BW0)                   ;ROUNDING
  A0F 04EE0 1E99002A (02049)        ADDB(BR1,BW1,TF)                ;SHAT(K)
  A10 04EE2 20A10039 (02050)        ADDL(BW2,1,TF)
  A11 04EE4 22A10039 (02051)        ADDL(BW2,1,TF)
  A12 04EE6 24290DB1 (02052)        SUBL(BR2,1),JUMPP(VTRAN)        ;LOOPING
                    (02053) *
                    (02054) *
                    (02055) *       SCALAR ADDRESSES
                    (02056) *
                    (02057) *       REGISTERS AFFECTED: BR0
```

```
                    (02058) *
A13 04EE8 2762####  (02059) VPCRCSS LOAD(BR2,MS$(1),L,TE)          ;ENCD. BETA
A14 04EEA 288A####2 (02060)         ADD(BR0,2,TF)                  ;RMS
A15 04EEC 2A8A####2 (02061)         ADD(BR0,2,TF)                  ;STATE VARIABLE
A16 04EEE 2C8A####2 (02062)         ADD(BR0,2,TF)                  ;UB*2**15/HL
A17 04EF0 2E8A####2 (02063)         ADD(BR0,2,TF)                  ;RMSMIN
A18 04EF2 308A####2 (02064)         ADD(BR0,2,TF)                  ;N-2.5
A19 04EF4 328A####2 (02065)         ADD(BR0,2,TF)                  ;ALAD
A1A 04EF6 348A####2 (02066)         ADD(BR0,2,TF)                  ;SMIN
A1B 04EF8 368A####2 (02067)         ADD(BR0,2,TF)                  ;UB
A1C 04EFA 388A####2 (02068)         ADD(BR0,2,TF)                  ;G
A1D 04EFC 3A8A####2 (02069)         ADD(BR0,2,TF)                  ;1-ALP
A1E 04EFE 3C8A####2 (02070)         ADD(BR0,2,TF)                  ;ALP
A1F 04F00 3E8A####2 (02071)         ADD(BR0,2,TF)                  ;AINV*(1/ALAD-1)
A20 04F02 408A####2 (02072)         ADD(BR0,2,TF)                  ;2**14
A21 04F04 428A####2 (02073)         ADD(BR0,2,TF)                  ;PITCH REPETITION SIZE
                    (02074) *
                    (02075) *       INITIALIZATION
                    (02076) *
                    (02077) *       REGISTER ASSIGNMENTS:
                    (02078) *
                    (02079) *       BR1 --- VHAT POINTER
                    (02080) *       BR2 --- Q BUFFER POINTER
                    (02081) *       BR3 --- SHAT(K-T-1) POINTER
                    (02082) *       BW0 --- 1023
                    (02083) *       BW1 --- 2048
                    (02084) *       BW3 --- SHAT(K-1)
                    (02085) *
                    (02086) *       REGISTERS AFFECTED: BR0,BR2,BR3,BW0,BW1,BW3
                    (02087) *
A22 04F06 4460101132 (02088)         LOAD(BR2,[1])                  ;
A23 04F08 4640####  (02089)         LOAD(BR0,MS$)                  ;
A24 04F0A 4822####  (02090)         SUB(BR2,MS$)                   ;ADDRESS POINTER FOR G BUFFER
A25 04F0C 4B2A####1 (02091)         ADD(BR2,1,TE)                  ;PITCH REPETITION CODE
          ####4F0F  (02092)         SH$RC=@L+1                     ;MODIFY SHAT ADDR.
A26 04F0E 4C7C8###  (02093)         LOAD(BR3,2048(2),S)            ;SHAT(K-1) POINTER (+N+G)
A27 04F10 4E31##17  (02094)         MOVB(BW3,BR3)                  ;SHAT(K-1)
          ####4F13  (02095)         T$RC=@L+1                      ;MODIFY T VALUE
A28 04F12 5404###32 (02096)         LOAD(BR0,50)                   ;T (RENEW EVERY BLOCK)
A29 04F14 5239##21  (02097)         SUBB(BR3,BR0)                  ;SHAT(K-T-1)
                    (02098) *
                    (02099) *       PITCH REPETITION MODE ?
                    (02100) *
                    (02101) *       REGISTERS AFFECTED: BR0,BR3,BW3
```

```
                    (0210) *
                    (0211) *        FLAGS SENSED: AF1,G3
                    (0212) *
A2A 04F16 54004E9 (0213) BWAIT    JUMPS(COUNT,AF1)                ;JUMP IF NO PITCH REPETITION
A2B 04F18 5600AA7 (0214)          JUMPC(BWAIT,G3)                 ;WAIT FOR APU SIGNAL
          0004F1B (0215)          IN$RC=#L+1                      ;INI PITCH REPETITION SIZE
A2C 04F1A 5840003B (0216)         LOAD(BR0,59)                    ;PITCH REPETITION SIZE
A2D 04F1C 5A3A0001 (0219) VLOP1   ADD(BR3,1)                      ;
A2E 04F1E 5C310039 (0218)         ADDL(BW3,1)                     ;
A2F 04F20 5E390000 (0211)         ANDB(BR3,BW0)                   ;
A30 04F22 60B900A2 (0212)         ADDB(BR3,BW1,TF)                ;SHAT(K-T)
A31 04F24 62310000 (0213)         ANDB(BW3,BW0)                   ;SHAT(K)
A32 04F26 6481002A (0214)         ADDB(BW3,BW1,TF)                ;
A33 04F28 66092D81 (0215)         SUBL(BR0,1),JUMPP(VLOP1)        ;LOOPING
                    (0216) *
                    (0217) *        INITIALIZATION
                    (0218) *
                    (0219) *        REGISTERS AFFECTED: BR1
                    (0220) *
                    (0221) *        FLAGS AFFECTED: AF1
                    (0222) *
A34 04F2A 68200029 (0223) COUNT   CLEAR(AF1)                      ;
          0004F2D (0224)          VH$RC=#L+1                      ;MODIFY VHAT ADDR.
A35 04F2C 6A5E0060 (0225)         LOAD(BR1,2048(3),S)             ;VHAT(K) POINTER
A36 04F2E 6C007060 (0226)         JUMP(ENDCK)                     ;
                    (0227) *
                    (0228) *        PRODUCING ADDRESSES FOR PREDICTOR WITH CIRCULAR BUFFER
                    (0229) *
                    (0230) *        REGISTERS AFFECTED: BR0,BR1
                    (0231) *
                    (0232) *        FLAGS AFFECTED: AF0
                    (0233) *
A37 04F30 6EC41000 (0234) VPRE1   LOAD(BR0,4096(2),L,TF)          ; **
A38 04F32 7020028 (0235)          CLEAR(AF0)                      ;
A39 04F34 7212001 (0236)          SUB(BR1,1)                      ;
A3A 04F36 7419000 (0237)          ANDB(BR1,BW0)
A3B 04F38 769900A2 (0238)         ADDB(BR1,BW1,TF)                ;
A3C 04F3A 7812001 (0239)          SUB(BR1,1)
A3D 04F3C 7A8A002 (0240)          ADD(BR0,2,TF)
A3E 04F3E 7C190000 (0241)         ANDB(BR1,BW0)
A3F 04F40 7E9900A2 (0242)         ADDB(BR1,BW1,TF)                ;
A40 04F42 8012001 (0243)          SUB(BR1,1)
A41 04F44 820A002 (0244)          ADD(BR0,2,TF)
A42 04F46 8419000 (0245)          ANDB(BR1,BW0)
```

```
PAGE   54:        MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980


  A43 04F48 8699002A (02146)         ADDB(BR1,BW1,TF)                ;
  A44 04F4A 88120001 (02147)         SUB(BR1,1)
  A45 04F4C 8A80A002 (02148)         ADD(BR0,2,TF)
  A46 04F4E 8C190000 (02149)         ANDB(BR1,BW0)
  A47 04F50 8E99002A (02150)         ADDB(BR1,BW1,TF)                ;
  A48 04F52 9801A004 (02151)         ADD(BR1,4)
  A49 04F54 9219FFFF (02152)         ANDB(BR1,BW0)
  A4A 04F56 9419002A (02153)         ADDB(BR1,BW1)
                     (02154) *
                     (02155) *       PRODUCING ADDRESSES FOR QUANTIZER
                     (02156) *
                     (02157) *       CONTROLLED BY HANDSHAKE SIGNAL (AF0,AF1) FROM APU PROGRAM
                     (02158) *
                     (02159) *       RREGISTERS AFFECTED: BR0
                     (02160) *
                     (02161) *       FLAGS AFFECTED: AF0,AF1
                     (02162) *       FLAGS SENSED: AF0,AF1
                     (02163) *
  A4B 04F58 96441006A (02164)        LOAD(BR0,4106(2),L)             ;ADDRESS OF EXPN(1) -6 **
  A4C 04F5A 9800051E9 (02165) VQ1    JUMPS(VQ3,AF1)                  ;FIND CORRESPONDING QUANTIZING LEVEL
  A4D 04F5C 9A004CA8 (02166)         JUMPC(VQ1,AF0)                  ;WAIT FOR SIGNAL
  A4E 04F5E 9C200028 (02167)         CLEAR(AF0)
  A4F 04F60 9E0A0004 (02168)         ADD(BR0,4)                      ;
  A50 04F62 A000C6C0 (02169)         JUMP(VQ1)
  A51 04F64 A280A002 (02170) VQ3     ADD(BR0,2,TF)                   ;EXPN(J)
  A52 04F66 A4200029 (02171)         CLEAR(AF1)
  A53 04F68 A680A002 (02172)         ADD(BR0,2,TF)                   ;OUT(J)
                     (02173) *
                     (02174) *       ADDRESSES OF SHAT(K-T),VHAT(K),SHAT(K)
                     (02175) *
                     (02176) *       REGISTERS AFFECTED: BR3,BW2,BW3
                     (02177) *
  A54 04F6A A83A0001 (02178)         ADD(BR3,1)                      ;
  A55 04F6C AA390000 (02179)         ANDB(BR3,BW0)
  A56 04F6E AC090028 (02180)         ADDB(BR3,BW1,TF)                ;SHAT(K-T)
  A57 04F70 AEA10013 (02181)         MOVB(BW2,BR1,TF)                ;VHAT(K)
  A58 04F72 B0310039 (02182)         ADDL(BW3,1)                     ;
  A59 04F74 B2310000 (02183)         ANDB(BW3,BW0)
  A5A 04F76 B4810028 (02184)         ADDB(BW3,BW1,TF)                ;SHAT(K)
                     (02185) *
                     (02186) *       ADAPTION
                     (02187) *
                     (02188) *       USING CIRCULAR BUFFER
                     (02189) *
```

```
                    (0219D) *       REGISTERS AFFECTED: BR0,BR1,BW2
                    (0219E) *
  A5B 04F78 B6C41000 (0219F) VADAP   LOAD(BR0,4096(2),L,TF)          ;STARTING ADDRESS OF A(I)  **
  A5C 04F7A B812D001 (021A0)         SUB(BR1,1)                     ;
  A5D 04F7C BA190000 (021A1)         ANDB(BR1,BW0)
  A5E 04F7E BC990D2A (021A2)         ADDB(BR1,BW1,TF)               ;
  A5F 04F80 BEA10011 (021A3)         MOVB(BW2,BR0,TF)               ;OUTPUT A(I)
  A60 04F82 C0120001 (021A4)         SUB(BR1,1)
  A61 04F84 C20A0002 (021A5)         ADD(BR0,2,TF)
  A62 04F86 C4190000 (021A6)         ANDB(BR1,BW0)
  A63 04F88 C6990D2A (021A7)         ADDB(BR1,BW1,TF)               ;
  A64 04F8A C8A10D3A (021A8)         ADDL(BW2,2,TF)                 ;
  A65 04F8C CA120001 (021A9)         SUB(BR1,1)
  A66 04F8E CC0A0002 (021AA)         ADD(BR0,2,TF)
  A67 04F90 CE190000 (021AB)         ANDB(BR1,BW0)
  A68 04F92 D0990D2A (021AC)         ADDB(BR1,BW1,TF)               ;
  A69 04F94 D2A10D3A (021AD)         ADDL(BW2,2,TF)
  A6A 04F96 D4120001 (021AE)         SUB(BR1,1)
  A6B 04F98 D60A0002 (021AF)         ADD(BR0,2,TF)
  A6C 04F9A D8190000 (021B0)         ANDB(BR1,BW0)
  A6D 04F9C DA990D2A (021B1)         ADDB(BR1,BW1,TF)               ;
  A6E 04F9E DCA10D3A (021B2)         ADDL(BW2,2,TF)
  A6F 04FA0 DE1A0005 (021B3)         ADD(BR1,5)
                    (021B4) *
                    (021B5) *       END OF RECEIVER 7
                    (021B6) *
                    (021B7) *       REGISTERS AFFECTED: BR2
                    (021B8) *
                    (021B9) *       FLAGS SENSED: AF0,AF3
                    (021BA) *
  A70 04FA2 E12A0001 (021BB) ENDCK   ADD(BR2,1,TE)                  ;INPUT L
  A71 04FA4 E2000020 (021BC) VLOP2   NOP                            ;
  A72 04FA6 E4003EE8 (021BD)         JUMPS(VPRE1,AF0)               ;JUMP IF NOT FINISH
  A73 04FA8 E6007AB (021BE)          JUMPC(VLOP2,AF3)               ;WAIT FOR APU SIGNAL
                    (021BF) *
                    (021C0) *       END OF RECEIVER 1
                    (021C1) *
                    (021C2) *       REGISTERS AFFECTED: BR0,BW3
                    (021C3) *
                    (021C4) *       FLAGS AFFECTED: AF3
                    (021C5) *
  A74 04FAA E8D20464 (021C6)         LOAD(BR1,SVT111S(1),L,TF)      ;1./(2.**15)
  A75 04FAC EA200028 (021C7)         CLEAR(AF3)                     ;
  A76 04FAE EC42044A (021C8)         LOAD(BR0,SVT99S(1),L)          ;
```

```
PAGE   56:         MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 3#, 198#

  A77 #4FB# EEB1##11 (#2234)          MOVB(BW3,BR#,TF)              ;STATE VARIABLE
  A78 #4FB2 F#B1##32 (#2235)          SUBL(BW3,2,TF)                ;RMS
  A79 #4FB4 F21A###2 (#2236)          ADD(BR1,2)
  A7A #4FB6 F531##13 (#2237)          MOVB(BW3,BR1,TE)              ;# OF SAMPLES
  A7B #4FB8 F62###31 (#2239)          CLEAR(RI)
  A7C #4FBA F8####2# (#2239)          NOP
  A7D #4FBC FA####6# (#224#)          JUMP(#)
                     (#2241) *
            ####4F#6 (#2242) VPCRCSA=#C                             ;CHAIN ANCHOR
                     (#2243) *
      #4FBE          (#2244)          END
                     (#2245) *
                     (#2246)          EVEN
                     (#2247) *
                     (#2248) *        STORAGE BLOCK FOR CONSTRUCTED INSTRUCTIONS
                     (#2249) *
      #4FBE ######## (#225#) VPCRCSI DATA    2F'#.#'
       ...
                     (#2251) *
            ####1## (#2252) VPCRCSZ=#L-VPCRCS                       ;COMPUTE MODULE SIZE
                     (#2253) *
                     (#2254) *        TOP OF EXEC
                     (#2255) *
                     (#2256) *         TOES=#L
                     (#2257) *         #L=TOESPRT
                     (#2258) *         ADDR    TOES(,BUS1S)
                     (#2259) *         END
                     (#226#) *
                     (#2261)          EVEN
                     (#2262)          EJECT
```

```
(02263) *
(02264) ;
(02265) ;
(02266) ;           ************************************
(02267) ;           *                                  *
(02268) ;           *    *** SOURCE CODE PROGRAM ***    *
(02269) ;           *                                  *
(02270) ;           ************************************
(02271) ;
(02272) ;                                   ARVIND S. ARORA
(02273) ;
(02274) ;      NAME: ENCDS       FCB: 110
(02275) ;
(02276) ;   THIS PROGRAM TAKES A BLOCK OF QUANTIZER OUTPUTS FROM THE PARC ALGORITHM
(02277) ;AND ENCODES THEM INTO A BIT STREAM FOR DIGITAL TRANSMISSION. EACH OUTPUT
(02278) ;BLOCK HAS 189 BITS CONSISTING OF 1 SYNC BIT, 6 BITS FOR T, 7 BITS FOR BETA,
(02279) ;7 BITS FOR 'PHONEY' BETA IF APPLICABLE, 18 BITS FOR PARITY CHECK, AND THE
(02280) ;REST (157 BITS) FOR QUANTIZER OUTPUT INFORMATION.
(02281) ;
(02282) ;
(02283) ;   ENCODER CONSTRAINTS:
(02284) ;
(02285) ;-- THE NUMBER OF QUANTIZER OUTPUTS IN THE INPUT BUFFER MUST GENERATE NO
(02286) ;MORE THAN 196 BITS.
(02287) ;-- NONE OF THE BUFFERS IN THIS PROGRAM MUST OCCUPY MEMORY LOCATIONS
(02288) ;HIGHER THAN 32K.
(02289) ;-- THERE ARE SEVERAL CONSTRAINTS ON THE SOURCE CODE:
(02290) ; 1. THE CODE FOR LEVEL 1 IS 0.
(02291) ; 2. THE CODE FOR PHONEY BETA IS 000,0000.
(02292) ; 3. RUN LENGTH IS FIXED AT 14. (UNLESS LOCATION LSRUN IS MODIFIED)
(02293) ; 4. THE CODE 111,1111 IS AN ALTERNATE NULL CODE.
(02294) ; 5. BECAUSE THE LSB OF THE CODE-WORD IS TRANSMITTED FIRST, THE RECEIVER
(02295) ;    GETS AN INVERTED CODE, EG. 110 --> 011. TO COMPENSATE FOR THIS,
(02296) ;    THE ENCODING TABLE MUST CONTAIN INVERTED CODES, RIGHT JUSTIFIED.
(02297) ; 6. LOCATION 0 IN THE ENCODING TABLE CONTAINS THE CODE FOR RUN LENGTH.
(02298) ; 7. LOCATION 2*12 IN THE ENCODING TABLE CONTAINS THE CODE FOR NULL CODE.
(02299) ;-- ALTHOUGH THE CODES FOR Q-LEVELS ARE INVERTED BY THE ALGORITHM, THE
(02300) ;CODES FOR T, BETA, PHONEY BETA ARE NOT.
(02301) ;-- THE VARIOUS INPUTS TO THIS PROGRAM NEED TO BE PROVIDED IN THE
(02302) ;FOLLOWING FORMATS:
(02303) ; 1. Q-LEVELS          N*2          FIXED FORMAT
(02304) ; 2. T                0<T<63       FIXED FORMAT (ENCODED T)
(02305) ; 3. BETA             0<BETA<96    FIXED FORMAT (ENCODED BETA)
(02306) ; 4. PHONEY BETA      -VE NO.      FIXED FORMAT
```

```
(0230) ; 5. END CODE        -VE 12      FIXED FORMAT
(0230) ;
(0230) ;
(0231)       EJECT
```

```
(02311) ;
(02312) ;   THE FUNCTION CONTROL BLOCK IS AS FOLLOWS:
(02313) ;
(02314) ;                  ----------------------------------------
(02315) ;                 |                    |                   |
(02316) ;                  ----------------------------------------
(02317) ;                 |    FCB # 110       |      ISTB         |
(02318) ;                  ----------------------------------------
(02319) ;                 |      OUTB          |      QHAT         |
(02320) ;                  ----------------------------------------
(02321) ;                 |      BETA (TABLE)  |      ENCT (TABLE) |
(02322) ;                  ----------------------------------------
(02323) ;                 |                    | SCAL ID(RC UPDATE)|
(02324) ;                  ----------------------------------------
(02325) ;                 *                    | INT SC ID(RC UPD) |
(02326) ;                  ----------------------------------------
(02327) ;
(02328) ;
(02329) ;
(02330) ; *** NOTE: ENABLE LINES WITH * IN COL. 1 TO ASSEMBLE THIS
(02331) ;           PROGRAM BY ITSELF.
(02332) ;
(02333) ;--- SYMBOL DEFINITION TO INTERFACE WITH SNAP-II EXEC. REL 3.05
(02334) ;
(02335) *       BCTSBA=$582
(02336) ;
(02337) *       FDTSUFCB=$8C4
(02338) ;
(02339) *       ISVTS=$502
(02340) ;
(02341) *       SYSSFLG=$1FFCE
(02342) ;
(02343) *       TOESOLD=$5000
(02344) *       TOESPTR=$288
(02345) ;
(02346) *       SVTS=$382
(02347) ;
(02348) ;--- TOP OF MEMORY POINTER
(02349) *#L=TOESPTR
(02350) *       ADDR    TOESNEW(,1)
(02351) ;
(02352) ;--- DISPATCH TABLE ENTRIES
(02353) *#L=FDTSUFCB
(02354) *       ADDR    ENCDS
```

```
                    (02355) ;
                    (02356) ;--- CSPU MODULE FOR THE ENCODER
                    (02357) ;
                    (02358) *0L=TOESOLD
                    (02359) ;--- I. BASE ADDRESS FOR 4 BUFFERS AND INDEX FOR T,BETA
                    (02360) ;
                    (02361) ;--- INDEX FOR T,BETA
                    (02362) ;
04FC2 702200FF      (02363) ENCDS   MOVWK   R2,R1,$00FF     ;SID FOR T,BETA
04FC4 E0205112      (02364)         MOVRM   R2,IIDSTB       SAVE IN LOC IIDSTB
                    (02365) ;
                    (02366) ;--- UPDATE RECEIVER
                    (02367) ;
                    (02368) ;        CALL    R7,RCS
                    (02369) ;
                    (02370) ;--- ADDRESS FOR QHAT
04FC6 2001          (02371)         NOP     2
04FC7 2000
                    (02372) ;
04FC8 2611          (02373)         INCR    R1,1
04FC9 706200FF      (02374)         MOVWK   R6,R1,$00FF     BID FOR QHAT
04FCB 86005DFF      (02375)         CALL    ,BASES          SUBROUTINE TO RETRIEVE BASE ADDR.
04FCD 844051A6      (02376)         MOVRML  R4,ADSQHAT      SAVE IN LOC ADSQHAT
                    (02377) ;
                    (02378) ;--- ADDRESS FOR OUTB
                    (02379) ;
04FCF 6062          (02380)         MOVWR   R6,R1
04FD0 3C68          (02381)         LRS     R6,8            BID FOR OUTB
04FD1 86005DFF      (02382)         CALL    ,BASES          SUB. TO GET BASE ADDR.
04FD3 844051B8      (02383)         MOVRML  R4,ADSOUTB      SAVE IN LOC ADSOUTB
04FD5 9C5000BD      (02384)         ADDIR   R5,$00BD        BASE ADDR + 189
04FD7 844051A0      (02385)         MOVRML  R4,ADSOUTT      SAVE IN LON ADSOUTT B. ADDR+189
                    (02386) ;
                    (02387) ;--- ADDRESS FOR ENCT, INDEXED BY R1
                    (02388) ;
04FD9 2611          (02389)         INCR    R1,1
04FDA 706200FF      (02390)         MOVWK   R6,R1,$00FF     BID FOR ENCT
04FDC 86005DFF      (02391)         CALL    ,BASES          SUBROUTINE TO GET BASE ADDR.
04FDE 96400002      (02392)         IORIR   R4,$0002        INSERT INDEX REG.1
04FE0 844051A0      (02393)         MOVRML  R4,ADSENCT      SAVE IN LOC ADSENCT
                    (02394) ;
                    (02395) ;--- ADDRESS FOR BETA, INDEXED BY R1
                    (02396) ;
04FE2 6062          (02397)         MOVWR   R6,R1
```

```
04FE3 3C68     (02398)          LRS      R6,8           BID FOR BETA
04FE4 8600050FF (02399)         CALL     ,BASE$         SUBROUTINE TO GET BASE ADDR.
04FE6 9640000002 (02400)        IORIR    R4,$0002       INSERT INDEX REG 1
04FE8 8440051AC (02401)         MOVRML   R4,AD$BETA     SAVE IN LOC AD$BETA
                (02402) ;
                (02403) ;--- II. INITIALIZE H-INDEX(R6), PARITY WORD(R7)
                (02404) ;
04FEA 9006000001 (02405)        MOVIR    R6,$0001       R6=1
04FEC 0D70     (02406)          CLR      R7             R7=0
                (02407) ;
                (02408) ;--- III. SYNC BIT; V$SYN
                (02409) ;
04FED 0C50     (02410)          CCR      R5             R5=-1($3FFFF)
04FEE F4500005113 (02411)       XORMR    R5,V$SYN       R5=COMPLEMENT(SYNC)
04FF0 8130004FF6 (02412)        JMP      EN$30,GEZ      IF SYNC=0, GO TO EN$30
04FF2 9001000001 (02413)        MOVIR    R1,$0001       FOR SYNC=1
04FF4 447C     (02414)          XORRR    R7,R6          PARITY UPDATE FOR 1
04FF5 1800     (02415)          SKP      0
04FF6 0D10     (02416) EN$30    CLR      R1             FOR SYNC=0
04FF7 E009051008 (02417)        MOVRM    R1,@AD$OUTB    OUTPUT SYNC
04FF9 E005005113 (02418)        MOVRM    R5,V$SYN       SAVE CURRENT SYNC
04FFB 2661     (02419)          INCR     R6,1           INCREMENT H-INDEX
04FFC E5000051009 (02420)       INCM     AD$OUTB+1      INCREMENT O-INDEX
                (02421) ;
                (02422) ;--- IV. OUTPUT T  (MSB FIRST)
                (02423) ;
04FFE F0105112 (02424)          MOVMR    R1,IID$TB      SID FOR T
05000 3A11     (02425)          LLS      R1,1
05001 F0052000382 (02426)       MOVMR    R5,SVT$(R1)    VALUE OF T (0-63)
05003 9004000005 (02427)        MOVIR    R4,$0005       (CODE LENGTH OF T)-1  (+VE INDEX)
                (02428) ;
05005 0D10     (02429) EN$40    CLR      R1
05006 2A5A     (02430)          SRBCL    5,R5           MSB OF T-CODE, SKIP IF 0
05007 2611     (02431)          INCR     R1,1           MAKE R1=1,IF MSB IS 1
05008 447C     (02432)          XORRR    R7,R6          UPDATE PARITY FOR 1
05009 E009051008 (02433)        MOVRM    R1,@AD$OUTB    OUTPUT
0500B 2661     (02434)          INCR     R6,1           INCREMENT H-INDEX
0500C E5000051009 (02435)       INCM     AD$OUTB+1      INCREMENT O-INDEX
0500E 3A51     (02436)          LLS      R5,1           SHIFT NEXT BIT TO MSB
0500F 8C4005005 (02437)         DJP      R4,EN$40       LOOP 6 TIMES
                (02438) ;
                (02439) ;--- V. CHECK FOR PHONEY BETA AND OUTPUT (MSB FIRST)
                (02440) ;
05011 F0D005106 (02441)         MOVMR    R5,@AD$QHAT    GET PHONEY BETA WORD
```

```
#5#13 8#2#5#1F (#2442)          JMP     ENS51,GTZ       >#, NO PHONEY BETA
                (#2443) ;                               PHONEY BETA =###,#### (7 BITS)
#5#15 9#4####6 (#2444)          MOVIR   R4,S####6       CODE LENGTH - 1      (+VE INDEX)
#5#17 #D1#     (#2445)          CLR     R1              R1=#
#5#18 E#9#51#8 (#2446) ENS5#    MOVRM   R1,@ADSOUTB     OUTPUT
#5#1A E5##51#9 (#2447)          INCM    ADSOUTB+1       INCREMENT O-INDEX
#5#1C 8C4#5#18 (#2448)          DJP     R4,ENS5#        LOOP 7 TIMES
                (#2449) ;
#5#1E 2667     (#245#)          INCR    R6,7            INCREMENT H-INDEX BY 7
#5#1F E5##51#7 (#2451) ENS51    INCM    ADSQHAT+1       INCREMENT I-INDEX
                (#2452) ;
                (#2453) ;--- VI. OUTPUT BETA (MSB FIRST)
                (#2454) ;
#5#21 F#5#5112 (#2455)          MOVMR   R5,IIDSTB
#5#23 3A51     (#2456)          LLS     R5,1
#5#24 2652     (#2457)          INCR    R5,2            SID FOR BETA
#5#25 F#1A#382 (#2458)          MOVMR   R1,SVTS(R5)     VALUE OF BETA (#-98)
#5#27 F#D#51#C (#2459)          MOVMR   R5,@ADSBETA     CODE FOR BETA
#5#29 9#4####6 (#246#)          MOVIR   R4,S####6       CODE LENGTH - 1      (+VE INDEX)
#5#2B #D1#     (#2461) ENS6#    CLR     R1
#5#2C 2A6A     (#2462)          SRBCL   6,R5            MSB OF BETA, SKIP IF #
#5#2D 2611     (#2463)          INCR    R1,1            R1=1,IF MSB IS 1
#5#2E 447C     (#2464)          XORRR   R7,R6           PARITY UPDATE FOR 1
#5#2F E#9#51#8 (#2465)          MOVRM   R1,@ADSOUTB     OUTPUT
#5#31 2661     (#2466)          INCR    R6,1            INCREMENT H-INDEX
#5#32 E5##51#9 (#2467)          INCM    ADSOUTB+1       INCREMENT O-INDEX
#5#34 3A51     (#2468)          LLS     R5,1            NEXT BIT IN MSB
#5#35 8C4#5#2B (#2469)          DJP     R4,ENS6#
                (#247#) ;
                (#2471) ;--- VII. FIRST,OUTPUT ANY EXTRA BITS FROM LAST BLOCK
                (#2472) ;
#5#37 2##1     (#2473)          NOP     2
#5#38 2###
#5#39 F#4#5114 (#2474)          MOVMR   R4,VSDEL        GET DELTA VALUE
#5#3B 8#3#5#4A (#2475)          JMP     ENS8#,LTZ       DEL<#, NO BITS LEFT OVER
#5#3D F#9#51#E (#2476) ENS7#    MOVMR   R1,@ADSOLDB     GET OLD OUTPUT
#5#3F 181#     (#2477)          SKP     EQZ
#5#4# 447C     (#2478)          XORRR   R7,R6           PARITY UPDATE FOR 1
#5#41 E#9#51#8 (#2479)          MOVRM   R1,@ADSOUTB     OUTPUT
#5#43 2661     (#248#)          INCR    R6,1            INCR H-INDEX
#5#44 E5##51#9 (#2481)          INCM    ADSOUTB+1       INCREMENT O-INDEX
#5#46 E5##51#F (#2482)          INCM    ADSOLDB+1       INCR OLD O-INDEX
#5#48 8C4#5#3D (#2483)          DJP     R4,ENS7#
                (#2484) ;
```

```
                    (0248S) ;--- VIII. OUTPUT REST OF INFORMATION BITS
0504A F090516 (0248S) ENS80    MOVMR    R1,@ADSQHAT    GET Q-OUT VALUE
0504C 9210002 (0248S)          CMPIR    R1,S0002       COMPARE WITH 1
0504E 8020554 (0248S)          JMP      ENS81,GTZ      >0, LEVEL OTHER THAN 1, OUT CODE
05050 8010566 (0248S)          JMP      ENS83,EQZ      =1, RUN-LENGTH CODE ?
05052 8000582 (0249S)          JMP      ENS90          <0, END CODE, GO FOR NULL OUTPUT
05054 C0C051A (0249S) ENS81    MOVMRL   R4,@ADSENCT    GET CODE & LENGTH - R5,R4
05056 501A001 (0249S) ENS82    MOVKR    R1,R5,S0001    LSB TO R1
05058 1810     (0249S)         SKP      EQZ
05059 447C     (0249S)         XORRR    R7,R6          PARITY UPDATE FOR 1
0505A E090518 (0249S)          MOVRM    R1,@ADSOUTB    OUTPUT
0505C 2661     (0249S)         INCR     R6,1           INCR H-INDEX
0505D E500519 (0249S)          INCM     ADSOUTB+1      INCR O-INDEX
0505F 9260039 (0249S)          CMPIR    R6,S0039       DELTA=(H-INDEX)-57
05061 1820     (0249S)         SKPL     LEZ
05062 8600500EF (0250S)        CALL     ,PAROUTS       IF 57 INFO BITS, PUT OUT 6 PAR. BITS
05064 3C51     (0250S)         LRS      R5,1           NEXT BIT TO LSB
05065 8C40056 (0250S)          DJP      R4,ENS82
                    (0250S) ;
05067 E500517 (0250S)          INCM     ADSQHAT+1      INCR I-INDEX
05069 8000504A (0250S)         JMP      ENS80
                    (0250S) ;
                    (0250S) ;--- RUN-LENGTH CODE
                    (0250S) ;
0506B F020005115 (0250S) ENS83 MOVMR    R2,LSRUN        1-COUNTER=-13     (-VE COUNTER)
0506D 0820     (0251S)         NEG      R2
0506E E500517 (0251S) ENS84    INCM     ADSQHAT+1      INCR I-INDEX
05070 F090516 (0252S)          MOVMR    R1,@ADSQHAT    GET NEXT Q-OUTPUT
05072 8030503A3 (0253S)        JMP      ENS88,LTZ      <0, END CODE, OUTPUT Q=1 BEFORE NULLS
05074 9210002 (0254S)          CMPIR    R1,S0002       CMPR WITH 1
05076 8020502 (0255S)          JMP      ENS86,GTZ      >1, NO RUN LENGTH
05078 8920506E (0256S)         IJN      R2,ENS84       LOOP 13 TIMES
                    (0257S) ;
                    (0258S) ;--- OUTPUT RUN LENGTH CODE  (AT LOC (ADSENCT)+0 )
                    (0259S) ;
0507A 0D10     (0260S)         CLR      R1             INDEX R1=0
0507B C0C051A (0261S)          MOVMRL   R4,@ADSENCT    GET RUN-LEN CODE, LENGTH - R5,R4
0507D 501A001 (0262S) ENS85    MOVKR    R1,R5,S0001    LSB TO R1
0507F 1810     (0263S)         SKP      EQZ
05080 447C     (0264S)         XORRR    R7,R6          PARITY UPDATE FOR 1
05081 E090518 (0265S)          MOVRM    R1,@ADSOUTB    OUTPUT
05083 2661     (0266S)         INCR     R6,1           INCR H-INDEX
05084 E500519 (0267S)          INCM     ADSOUTB+1      INCR O-INDEX
05086 9260039 (0268S)          CMPIR    R6,S0039       DEL=H-IND -1
```

```
0508 1820      (0529)          SKPL    LEZ
0509 8605050EF (0530)          CALL    ,PAROUTS        IF 57 INFO BITS, OUT 6 PARITY BITS
050B 3C51      (0531)          LRS     R5,1            NEXT BIT TO LSB
050C 8C4050 7D (0532)          DJP     R4,ENS05
               (0533) ;
050E E5000517  (0534)          INCM    ADSQHAT+1       INCR I-INDEX
0510 80000504A (0535)          JMP     ENS00
               (0536) ;
               (0537) ;--- NO RUN-LENGTH CODE
               (0538) ;
0512 FC200115  (0539) ENS06    ADDMR   R2,LSRUN        1-CNTR=1-CNTR+13  (0 OF 0'S,+VE INDEX)
0514 0D50      (0540)          CLR     R5              CODE FOR Q=1 IS 0, R5=0
0515 E0D00518  (0541) ENS07    MOVRM   R5,@ADSOUTB     OUTPUT
0517 2661      (0542)          INCR    R6,1            INCR H-INDEX
0518 E5000519  (0543)          INCM    ADSOUTB+1       INCR O-INDEX
051A 92600039  (0544)          CMPIR   R6,$0039        DEL=H-IND -57
051C 1820      (0545)          SKPL    LEZ
051D 8605050EF (0546)          CALL    ,PAROUTS        IF 57 INFO BITS, OUTPUT 6 PARITY BITS
051F 8C205095  (0547)          DJP     R2,ENS07
               (0548) ;
0521 8000504   (0549)          JMP     ENS01
               (0550) ;
               (0551) ;--- PUT OUT 1'S COLLECTED IN RUN LENGTH PROCEDURE, END-CODE
               (0552) ;
0523 FC200115  (0553) ENS08    ADDMR   R2,LSRUN        1-CNTR=1-CNTR +13  (0 OF 0'S, +VE INDEX)
0525 0D50      (0554)          CLR     R5              Q=1, CODE IS 0,R5=0
               (0555) ENS09
0526 E0D00518  (0556)          MOVRM   R5,@ADSOUTB     OUTPUT
0528 2661      (0557)          INCR    R6,1            INCR H-INDEX
0529 E5000519  (0558)          INCM    ADSOUTB+1       INCR O-INDEX
052B 92600039  (0559)          CMPIR   R6,$0039        DEL=H-IND -57
052D 1820      (0560)          SKPL    LEZ
052E 8605050EF (0561)          CALL    ,PAROUTS        IF 57 INFO BITS, PUT OUT 6 PARITY BITS
0530 8C205026  (0562)          DJP     R2,ENS09
               (0563) ;
               (0564) ;--- IX. END-PROCEDURE  (NULL CODE)
               (0565) ;
0532 F040051 9 (0566) ENS90    MOVMR   R4,ADSOUTB+1    MOVE O-INDEX TO R3
0534 FE400111  (0567)          SUBMR   R4,ADSOUTT+1    DIFF=O-INDEX -189
0536 8130500E4 (0568)          JMP     ENS102,GEZ      IF DIFFERENCE +VE, READY TO QUIT
               (0569) ;
0538 9010000 18 (0570)         MOVIR   R1,$0018        INDEX FOR NULL CODE (12*2)
053A C0C05100A (0571)          MOVMRL  R4,@ADSENCT     NULL CODE,LENGTH IN R5,R4
053C 501A0001  (0572) ENS91    MOVKR   R1,R5,$0001     LSB TO R1
```

```
#5#BE 181#      (#2573)           SKP       EQZ
#5#BF 447C      (#2574)           XORRR     R7,R6           PARITY UPDATE FOR 1
#5#C# E#9#51#8 (#2575)           MOVRM     R1,@AD$OUTB     OUTPUT
#5#C2 2661      (#2576)           INCR      R6,1            INCR H-INDEX
#5#C3 E5##51#9 (#2577)           INCM      AD$OUTB+1       INCR O-INDEX
#5#C5 926###39 (#2578)           CMPIR     R6,$##39        DEL=H-IND -57
#5#C7 182#      (#2579)           SKPL      LEZ
#5#C8 86##5#EF (#258#)           CALL      ,PAROUT$        IF 57 INFO BITS, PUT OUT 6 PARITY BITS
#5#CA 3C51      (#2581)           LRS       R5,1            NEXT BIT TO LSB
#5#CB 8C4#5#BC (#2582)           DJP       R4,EN$91
                (#2583) ;
                (#2584) ;--- X. IF MORE SPACE, PAD IN WITH 1'S AFTER NULL CODE
                (#2585) ;
#5#CD F#4#51#9 (#2586) EN$1##    MOVMR     R4,AD$OUTB+1    MOVE O-IND TO R3
#5#CF FE4#5111 (#2587)           SUBMR     R4,AD$OUTT+1    DIFF=O-IND -189
#5#D1 813#5#E4 (#2588)           JMP       EN$1#2,GEZ      IF DIFF .GE. #, READY TO QUIT
                (#2589) ;
                (#259#) ;                                  (-VE COUNTER)
#5#D3 9#5#####1 (#2591)           MOVIR     R5,$###1        PADDING IN WITH 1'S
#5#D5 E#D#51#8 (#2592) EN$1#1    MOVRM     R5,@AD$OUTB     OUTPUT
#5#D7 447C      (#2593)           XORRR     R7,R6           PARITY UPDATE FOR 1
#5#D8 2661      (#2594)           INCR      R6,1            INCR H-INDEX
#5#D9 E5##51#9 (#2595)           INCM      AD$OUTB+1       INCR O-INDEX
#5#DB 926###39 (#2596)           CMPIR     R6,$##39        DEL=H-IND -57
#5#DD 812#5#E2 (#2597)           JMP       EN$111,LEZ
#5#DF 86##5#EF (#2598)           CALL      ,PAROUT$        IF 57 INFO BITS, OUTPUT 6 PARITY
#5#E1 2646      (#2599)           INCR      R4,6            INCR INDX R3 6 FOR PARITY BITS
#5#E2 894#5#D5 (#26##) EN$111    IJN       R4,EN$1#1
                (#26#1) ;
                (#26#2) ;--- COMPUTE DELTA, UPDATE OLD ADDRESS AND QUIT
                (#26#3) ;
#5#E4 C#4#511# (#26#4) EN$1#2    MOVMRL    R4,AD$OUTT      ADDR FOR EXTRA BITS (O-ADDR +189)
#5#E6 F#3#51#9 (#26#5)           MOVMR     R3,AD$OUTB+1    O-INDEX TO R3
#5#E8 4E3A      (#26#6)           SUBRR     R3,R5           DIFF=O-INDEX -189
#5#E9 2731      (#26#7)           DECR      R3,1            STORE DIFF-1
#5#EA E#3#5114 (#26#8)           MOVRM     R3,VSDEL
#5#EC 844#51#E (#26#9)           MOVRML    R4,AD$OLDB      SAVE OLD ADDR
                (#261#) ;
                (#2611) ;
#5#EE #E7#      (#2612)           RETURN
                (#2613) ;
                (#2614) ;
                (#2615) ;--- SUBROUTINE  PAROUT$
                (#2616) ;
```

```
                    (0617) ;     - TO OUTPUT 6 PARITY BITS (MSB FIRST)
                    (0618) ;     - INITIALIZE PARITY CHECK WORD (R7) TO 0
                    (0619) ;     - INITIALIZE H-INDEX (R6) TO 1
                    (0620) ;
0500EF 9030FFF5     (0621) PAROUT$  MOVIR    R3,$FFF5        TO MOVE OUT 6 BITS (+VE INDEX)
0500F1 5006E002     (0622) PAR$1    MOVKR    R6,R7,$E002     6TH BIT TO R6
0500F3 3C65         (0623)          LRS      R6,5            SHIFT BIT TO LSB
0500F4 E8E051F8     (0624)          MOVRM    R6,@AD$OUTB     OUTPUT
0500F6 E5FF51F9     (0625)          INCM     AD$OUTB+1       INCREMENT O-INDEX
0500F8 3A71         (0626)          LLS      R7,1            NEXT BIT TO 6TH POSITION
0500F9 0C3F50F1     (0627)          DJP      R3,PAR$1
                    (0628) ;
0500FB 9006FFF1     (0629)          MOVIR    R6,$FFF1        INITIALIZE R6 TO 1
0500FD 0D70         (0630)          CLR      R7              INITIALIZE R7 TO 0
                    (0631) ;
0500FE 0E70         (0632)          RETURN
                    (0633) ;
                    (0634) ;
                    (0635) ;--- SUBROUTINE  BASES
                    (0636) ;
                    (0637) ;     - TO GET BASE ADDR AND MEM BUS FOR BUFFER (NOT INDEX REGISTER)
                    (0638) ;     - R6 = BID
                    (0639) ;     - R4,R5 = BASE ADDRESS
                    (0640) ;
0500FF 3A61         (0641) BASES    LLS      R6,1            INDEX=BID*2
050100 C04C0582     (0642)          MOVMRL   R4,BCT$BA(R6)     GET BASE ADDR FROM BCT
050102 9A4FFFF6     (0643)          ANDIR    R4,$FFF6          KEEP ONLY BUS ID
050104 3A43         (0644)          LLS      R4,3              MOVE BUS ID TO BITS 4,5
050105 0E70         (0645)          RETURN
                    (0646) ;
                    (0647) ;
                    (0648) ;--- LEAVE SPACE FOR VARIOUS VARIABLES
                    (0649) ;
                    (0650)          EVEN
050106 00000000     (0651) AD$QHAT  DATA     F'0.0'
050108 00000000     (0652) AD$OUTB  DATA     F'0.0'
05010A 00000000     (0653) AD$ENCT  DATA     F'0.0'
05010C 00000000     (0654) AD$BETA  DATA     F'0.0'
05010E 00000000     (0655) AD$OLDB  DATA     F'0.0'
050110 00000000     (0656) AD$OUTT  DATA     F'0.0'
050112 0000         (0657) IID$TB   DATA     $0000
050113 0001         (0658) V$SYN    DATA     $0001
050114 FFFF         (0659) VSDEL    DATA     $FFFF
050115 0000         (0660) L$RUN    DATA     $0000
```

```
(#2661) ;
(#2662) ;
(#2663) ;--- UPDATE TOP OF MEMORY
(#2664) ;
(#2665) *         TOESNEW=#L
(#2666) *         END
(#2667)           EVEN
(#2668)           EJECT
```

```
(#2669) ;
(#2670) ;
(#2671) ;
(#2672) ;            ********************************
(#2673) ;            *                              *
(#2674) ;            *   *** PROGRAM  TRBUF ***     *
(#2675) ;            *                              *
(#2676) ;            ********************************
(#2677) ;
(#2678) ;           PROG:TRBDC                    ARVIND S. ARORA
(#2679) ;
(#2680) ;           FCB = #112
(#2681) ;
(#2682) ;     THIS PROGRAM TRANSFERS ELEMENTS FROM A DOUBLE BUFFER (100 LONG
(#2683) ; EACH) TO A CIRCULAR BUFFER (1024 LONG). THIS IS USEFUL IN TESTING
(#2684) ; THE PARC ALGORITHM WITHOUT THE USE OF THE IOS2. THIS PROGRAM
(#2685) ; ESSENTIALLY SIMULATES THE IOS2 IN THE CSPU. SINCE THIS PROGRAM
(#2686) ; ADDS TO THE CSPU TIME, WHICH IS ALREADY AT PREMIUM, THE SAMPLING
(#2687) ; RATE MUST BE LOWERED BY ABOUT 33 PERCENT.
(#2688) ;
(#2689)       EJECT
```

```
(#269#) ;
(#2691) ;     THE FUNCTION CONTROL BLOCK IS AS FOLLOWS:
(#2692) ;
(#2693) ;                  ------------------------------------
(#2694) ;                 |                  |                  |
(#2695) ;                  ------------------------------------
(#2696) ;                 |  FCB # 112       |  INIT            |
(#2697) ;                  ------------------------------------
(#2698) ;                 |  DBUF2           |  DBUF1           |
(#2699) ;                  ------------------------------------
(#27##) ;                 |                  |  CBUF            |
(#27#1) ;                  ------------------------------------
(#27#2) ;                 |                  |                  |
(#27#3) ;                  ------------------------------------
(#27#4) ;                 |                  |                  |
(#27#5) ;                  ------------------------------------
(#27#6) ;
(#27#7) ;
(#27#8) ;
(#27#9) ; *** NOTE: ENABLE LINES WITH * IN COLUMN 1 TO ASSEMBLE THIS
(#271#) ;           ROUTINE ALONE.
(#2711) ;
(#2712) ;--- SYMBOL DEFINITION TO INTERFACE WITH SNAP-II EXEC. REL 3.#5
(#2713) ;
(#2714) *         $CT$BA=$582
(#2715) *         FDT$UFCB=$8C8
(#2716) *         I$VT$=$5#2
(#2717) *         SYS$FLGS=$1FFCE
(#2718) *         TOE$OLD=$51B#
(#2719) *         TOE$PTR=$288
(#272#) ;
(#2721) *#L=TOE$PTR
(#2722) *         ADDR    TOE$NEW(,1)
(#2723) ;
(#2724) ;--- DISPATCH TABLE ENTRY
(#2725) ;
(#2726) *#L=FDT$UFCB
(#2727) *         ADDR    TR$DC
(#2728) ;
(#2729) ;--- CSPU MODULE FOR BUFFER TRANSFER,  DOUBLE BUFF. --> CIRCULAR BUFF.
(#273#) ;
(#2731) *#L=TOE$OLD
(#2732) ;
(#2733) ;--- CHECK INIT. SCALAR SELECT
```

```
                    (#2734) ;
#5116 7#72##FF      (#2735) TR$DC   MOVWK   R7,R1,$FF
                    (#2736) ;       MOVLM   $27,SYS$FLGS        TIMING INSTRUCTION
#5118 2611          (#2737)         INCR    R1,1
#5119 F#6E#5#2      (#2738)         MOVMR   R6,ISVT$(R7)
#511B 8#2#5135      (#2739)         JMP     TR$2#,GTZ           >#,NORMAL OPERATION BUF 1
#511D 8#3#512A      (#274#)         JMP     TR$1#,LTZ           <#,NORMAL OPERATION BUF 2
                    (#2741) ;                                   ;=#,INITIALIZATION
                    (#2742) ;
#511F #C6#          (#2743)         CCR     R6                  SET SCALAR SELECT TO -1
                    (#2744) ;
#512# 2611          (#2745)         INCR    R1,1
#5121 6#22          (#2746)         MOVWR   R2,R1
#5122 3A21          (#2747)         LLS     R2,1
#5123 C#44#582      (#2748)         MOVMRL  R4,BCT$BA(R2)       GET BASE ADDR FOR CIRC. BUFF
#5125 E#5#514F      (#2749)         MOVRM   R5,ADOUT            SAVE ADDR
                    (#275#) ;
#5127 CC##515F      (#2751)         MOVZM   IND$OUT
#5129 2711          (#2752)         DECR    R1,1
                    (#2753) ;
                    (#2754) ;
#512A #B6#          (#2755) TR$1#   NEG     R6                  SET SID SELECT
#512B E#6E#5#2      (#2756)         MOVRM   R6,ISVT$(R7)        SAVE
                    (#2757) ;
#512D 6#72          (#2758)         MOVWR   R7,R1               GET BID FOR DOUBLE BUFF 1
#512E 3C77          (#2759)         LRS     R7,7
#512F C#4E#582      (#276#)         MOVMRL  R4,BCT$BA(R7)       GET BASE ADDR
#5131 E#5#514C      (#2761)         MOVRM   R5,ADIN             SAVE BASE ADDR
#5133 8###513F      (#2762)         JMP     TR$25
                    (#2763) ;
                    (#2764) ; ----- FOR BUFFER 2
                    (#2765) ;
#5135 #B6#          (#2766) TR$2#   NEG     R6                  COMPLEMENT SID SELECT
#5136 E#6E#5#2      (#2767)         MOVRM   R6,ISVT$(R7)        SAVE
                    (#2768) ;
#5138 7#72##FF      (#2769)         MOVWK   R7,R1,$FF           GET BID FOR DOUBLE BUFF 2
#513A 3A71          (#277#)         LLS     R7,1
#513B C#4E#582      (#2771)         MOVMRL  R4,BCT$BA(R7)       GET BASE ADDR
#513D E#5#514C      (#2772)         MOVRM   R5,ADIN             SAVE BASE ADDR
                    (#2773) ;
                    (#2774) ;
#513F F#7#516#      (#2775) TR$25   MOVMR   R7,SIZE$            GET BUFFER SIZE
#5141 2771          (#2776)         DECR    R7,1                SET UP COUNTER
                    (#2777) ;
```

```
                    (02778) ;
                    (02779) ; ----- GET MODE FOR IOS FROM INIT+1
                    (02780) ;
 0142 2711          (02781)          DECR    R1,1
 0143 7052FF        (02782)          MOVWK   R5,R1,$FF          ISID FOR INIT
 0145 2651          (02783)          INCR    R5,1               ISID FOR IOS MODE
 0146 F05A052       (02784)          MOVMR   R5,ISVT$(R5)       GET IOS MODE WORD
                    (02785) ;
                    (02786) ; ----- SET UP IN/OUT ADDR INDICES
                    (02787) ;
 0148 0D30          (02788)          CLR     R3                 INP IND
 0149 F020515F      (02789)          MOVMR   R2,IND$OUT         OUT IND
                    (02790) ;
                    (02791) ;
                    (02792) ;--- TRANSFER LOOP
                    (02793) ;
      014C          (02794) ADIN=@L+1
 014B F066515A      (02795) TR$3     MOVMR   R6,AD$IN(R3)
 014D 466A          (02796)          IORRR   R6,R5               ADD IN IOS MODE
      014F          (02797) ADOUT=@L+1
 014E E064515C      (02798)          MOVRM   R6,AD$OUT(R2)
                    (02799) ;
 0150 2631          (02800)          INCR    R3,1               UPDATE INP COUNTER
 0151 2621          (02801)          INCR    R2,1                      OUT COUNTER
 0152 9A203FF       (02802)          ANDIR   R2,$03FF                  MOD 1024
                    (02803) ;
 0154 8C70148       (02804)          DJP     R7,TR$30
                    (02805) ;
                    (02806) ;
 0156 E020515F      (02807)          MOVRM   R2,IND$OUT         SAVE OUT INDEX
                    (02808) ;
                    (02809) ;        MOVLM   $7,SYS$FLGS        TIMING INSTRUCTION,  G3
 0158 0E70          (02810)          RETURN
 0159 0000          (02811)          EVEN
                    (02812) ;
                    (02813) ;--- SPACE FOR DATA
                    (02814) ;
 015A 00000000      (02815) AD$IN    DATA    F'0.0'
 015C 00000000      (02816) AD$OUT   DATA    F'0.0'
 015E 0000          (02817) IND$IN   DATA    $0000
 015F 0000          (02818) IND$OUT  DATA    $0000
 0160 00BD          (02819) SIZE$    DATA    $BD
                    (02820) ;
                    (02821) ;--- UPDATE TOP OF MEMORY
```

```
                    (#2822) ;
                    (#2823) *         TOESNEW=#L
                    (#2824) *         END
      #5161 ####    (#2825)           EVEN
                    (#2826)           EJECT
```

```
(#2827) ;
(#2828) ;
(#2829) ;
(#283#) ;          ****************************************
(#2831) ;          *                                      *
(#2832) ;          *    *** SOURCE DECODE PROGRAM ***     *
(#2833) ;          *                                      *
(#2834) ;          ****************************************
(#2835) ;
(#2836) ;                                        ARVIND S. ARORA
(#2837) ;
(#2838) ;        NAME: DCDR$              FCB: 111
(#2839) ;
(#284#) ;    THIS PROGRAM TAKES FRAMES OF 189 BITS (LSB IN THE 16 BIT WORDS
(#2841) ;OUTPUT BY THE IOS2), SYNCHRONIZES ON THE FRAME, AND DECODES THE BITS
(#2842) ;INTO A BLOCK OF QUANTIZER LEVELS TO BE USED BY THE RECEIVER OF THE
(#2843) ;PARC ALGORITHM.
(#2844) ;    DEPENDING ON THE VALUES OF 2 FUNCTION SELECT INDICATORS, IT SELECTS
(#2845) ;EITHER TO DECODE, OR SYNCHRONIZE, OR INITIALIZE AND SYNCHRONIZE. IT
(#2846) ;ALSO DECIDES WHICH OF THE TWO Q-OUT BUFFERS TO FILL UP.
(#2847) ;    THE Q-OUT BUFFER HAS THE FOLLOWING FORMAT: PHONEY BETA INDICATOR
(#2848) ;(-1 MEANS PHONEY BETA PRESENT), QUANTIZER LEVELS (1 TO 11, AND A RUN-
(#2849) ;LENGTH CODE TO BE DECODED TO 14 LEVEL 1 OUTPUTS), AND AN END-OF-BLOCK
(#285#) ;CODE (-VE 12).
(#2851) ;    EACH TIME THE PROGRAM IS STOPPED, THE MODULE MUST BE RE-LOADED
(#2852) ;OR LOCATIONS P$BASE,L$RUN RESET TO ENSURE PROPER INITIALIZATION.
(#2853) ;
(#2854) ;    THIS PROGRAM ALSO SIMULATES THE STATE OF THE SHAT BUFFER IN THE
(#2855) ;RECEIVER TO ENSURE THE BUFFER DOES NOT UNDER-FLOW OR OVER-FLOW.
(#2856) ;
(#2857) ;
(#2858) ;    CONSTRAINTS:
(#2859) ;
(#286#) ; 1)THE CONSTRAINTS ON THE SOURCE CODE ARE THE SAME AS THOSE ON THE
(#2861) ;   ENCODER.
(#2862) ; 2)THE OUTPUT OF THE DECODER HAS THE SAME FORM AS THE INPUT OF THE
(#2863) ;   ENCODER.
(#2864) ; 3)THE INPUT OF THIS PROGRAM IS A CIRCULAR BUFFER OF LENGTH 1#24.
(#2865) ; 4)ALL THE BUFFERS USED IN THIS PROGRAM MUST RESIDE ON BUS 1.
(#2866) ;
(#2867)     EJECT
```

```
(#2868) ;
(#2869) ;    THE FUNCTION CONTROL BLOCK HAS THE FOLLOWING FORMAT:
(#287#) ;
(#2871) ;                  ----------------------------------------
(#2872) ;                 |                        |               |
(#2873) ;                  ----------------------------------------
(#2874) ;                 |    FCB # 111           |    ISELECT    |
(#2875) ;                  ----------------------------------------
(#2876) ;                 |       OOB2             |       OOB1    |
(#2877) ;                  ----------------------------------------
(#2878) ;                 |       ITB2             |       ITB1    |
(#2879) ;                  ----------------------------------------
(#288#) ;                 |      DECT (TABLE)      |      DECB (TABLE)  |
(#2881) ;                  ----------------------------------------
(#2882) ;                 |       IGAP             |       ICB     |
(#2883) ;                  ----------------------------------------
(#2884) ;
(#2885) ;
(#2886) ; *** NOTE: ENABLE ALL LINES WITH * IN COLUMN TO ASSEMBLE THIS
(#2887) ;           PROGRAM ALONE.
(#2888) ;
(#2889) ;--- SYMBOL DEFINITION TO INTERFACE WITH SNAP-II EXEC. REL 3.#5
(#289#) ;
(#2891) *        BCTS8A=$582           BASE OF SCALAR TABLE
(#2892) ;
(#2893) *        FDTSUFCB=S8C6         FCB # 111
(#2894) ;
(#2895) *        ISVTS=$5#2            BASE OF INTEGER SCALAR TABLE
(#2896) ;
########9 (#2897)          NOSMAX=$9             (NO OF WORDS IN SYNC CORR. COMPUTATION)
(#2898) ;
(#2899) *        SYSSFLG=$1FFCE        SYSTEM FLAG LOCATION
(#29##) ;
(#29#1) *        TOESOLD=$52##        OLD TOP OF MEMORY
(#29#2) *        TOESPTR=$288         TOP OF MEMORY POINTER
(#29#3) ;
(#29#4) *        SVTS=S382             BASE OF SCALAR TABLE
(#29#5) ;
(#29#6) ;--- TOP OF MEMORY POINTER
(#29#7) ;
(#29#8) *#L=TOESPTR
(#29#9) ;
(#291#) *        ADDR    TOESNEW(,1)
(#2911) ;
```

```
                    (0292) ;--- DISPATCH TABLE ENTRIES
                    (0293) ;
                    (0294) *@L=FDTSUFCB
                    (0295) ;
                    (0296) *         ADDR    DCDRS
                    (0297) ;
                    (0298) ;
                    (0299) ;--- CSPU MODULE FOR THE DECODER
                    (0300) ;
                    (0301) *@L=TOESOLD
                    (0302) ;
                    (0303) ;    FIRST DECIDE WHICH OPERATIONS ARE TO BE DONE
                    (0304) ;
0162 E010548A4      (0305) DCDRS    MOVRM   R1,BASFCB       STACK FCB POINTER
0164 2001           (0306)          NOP     2               SAVE SPACE FOR TIMING INSTRUCTION
0165 2000
                    (0307) ;
                    (0308) ;--- USE FUNCTION SELECT INDICATOR TO DECIDE OPERATION
                    (0309) ;
0166 7022000FF      (0310)          MOVWK   R2,R1,$00FF     GET SID OF ISL
0168 E020548B6      (0311)          MOVRM   R2,SIDSISL      SAVE SID
016A F034005A2      (0312)          MOVMR   R3,ISVTS(R2)    GET ISL1
016C 8020005314     (0313)          JMP     DECDS,GTZ       ISL1 > 0, DECODER
016E 8030051C0      (0314)          JMP     SYNCS,LTZ       ISL1 < 0, SYNCHRONIZE
                    (0315) ;                                ;ISL1 = 0, INITIALIZE & SYNCHRONIZE
                    (0316) ;
                    (0317) ;
                    (0318) ;              *** INITIALIZATION ***
                    (0319) ;
                    (0320) ;
                    (0321) ; 1) SET ISL1 TO -1
                    (0322) ;
                    (0323)          EVEN
017000C30           (0324) INITS    CCR     R3              SET R3=-1
0171 E034005A2      (0325)          MOVRM   R3,ISVTS(R2)    SET ISL1 TO -1
                    (0326) ;
                    (0327) ; 2) SET MAX CORRELATION INDEX WORDS (10) TO -1
                    (0328) ;
0173 0C500          (0329)          CCR     R5              R5=-1
0174 904000009      (0330)          MOVIR   R4,NOSMAX       INDEX=9 (FOR 10 WORDS)
0176 E050548B       (0331) INS20    MOVRM   R5,PSMAX(R4)    MAX CORR WORD = -1
0178 8C4005176      (0332)          DJP     R4,INS20
                    (0333) ;
                    (0334) ; 3) GET BASE ADDRESS OF Q-BUFFER1 & SAVE IN BASOQB
```

```
                    (0256) ;           ( TO BE USED FOR STORING SYNC CORRELATION VALUES )
                    (0256) ;           ( INDEXED BY R2 )
                    (0257) ;
    0517A 2611      (0258)           INCR    R1,1
    0517B 7022FF00  (0259)           MOVWK   R2,R1,$00FF      BID OF OQB1
    0517D 3A21      (0260)           LLS     R2,1             BID*2
    0517E C034B582  (0261)           MOVMRL  R3,BCTSBA(R2)    GET BASE ADDR WORD FOR OQB1
    05180 9030000E  (0262)           MOVIR   R3,$E            BUS 1, IND REG R2
    05182 8430B4AE  (0263)           MOVRML  R3,BASOQB        SAVE BASE ADDR
                    (0264) ;
                    (0265) ;--- UPDATE PROGRAM LOCATIONS REFERRING TO OQB
                    (0266) ;
    05184 E040B194  (0267)           MOVRM   R4,OQB10
    05186 E040B1E2  (0268)           MOVRM   R4,OQB11
    05188 E040B1EA  (0269)           MOVRM   R4,OQB12
    0518A E040B1F0  (0270)           MOVRM   R4,OQB13
    0518C E040B1F8  (0271)           MOVRM   R4,OQB14
    0518E E040B200  (0272)           MOVRM   R4,OQB15
                    (0273) ;
                    (0274) ; 4) INITIALIZE OQB BUFFER TO 0
                    (0275) ;
    05190 902000EF  (0276)           MOVIR   R2,$EF           R2=399
    05192 0D50      (0277)           CLR     R5
          0000B194  (0278) OQB10=@L+1
    05193 E0B5AE00  (0279) INS40     MOVRM   R5,BASOQB(R2)    INIT OQB TO 0
    05195 8C2B5193  (0280)           DJP     R2,INS40
                    (0281) ;
                    (0282) ; 5) GET BASE ADDR OF INPUT CIRCULAR BUFFER, IND REG R3, SIZE=1024
                    (0283) ;
    05197 2613      (0284)           INCR    R1,3
    05198 7022FF00  (0285)           MOVWK   R2,R1,$00FF      BID OF ICB
    0519A 3A21      (0286)           LLS     R2,1             BID*2
    0519B C034B582  (0287)           MOVMRL  R3,BCTSBA(R2)    GET BASE ADDR WORDS FOR ICB
    0519D 9030000E  (0288)           MOVIR   R3,$E            BUS 1, IND REG R3
    0519F 8430B4AC  (0289)           MOVRML  R3,BASICB        SAVE BASE ADDR
                    (0290) ;
                    (0291) ;--- MODIFY PROGRAM LOCATIONS REFERRING TO ICB
                    (0292) ;
    051A1 E040B1DD  (0293)           MOVRM   R4,ICB10
    051A3 E040B297  (0294)           MOVRM   R4,ICB11
    051A5 E040B2A9  (0295)           MOVRM   R4,ICB12
    051A7 E040B2C8  (0296)           MOVRM   R4,ICB13
    051A9 E040B2DD  (0297)           MOVRM   R4,ICB14
                    (0298) ;
```

```
      0518A E040E355 (02999)          MOVRM   R4,ICB20
      0518D E040E363 (03000)          MOVRM   R4,ICB21
      0518F E040E376 (03001)          MOVRM   R4,ICB22
      05191 E040E383 (03002)          MOVRM   R4,ICB23
      05193 E040E3C1 (03003)          MOVRM   R4,ICB24
      05195 E040E3D6 (03004)          MOVRM   R4,ICB25
      05197 E040E400 (03005)          MOVRM   R4,ICB26
      05199 E040E435 (03006)          MOVRM   R4,ICB27
                     (03007) ;
                     (03008) ; 6) SAVE GAP SIZE
                     (03009) ;
      0519B 6022     (03010)          MOVWR   R2,R1
      0519C 3C20     (03011)          LRS     R2,8            GET GAP SIZE
      0519D E020E4D5 (03012)          MOVRM   R2,VSGAP        SAVE GAP SIZE
                     (03013) ;
                     (03014) ;
                     (03015) ;
                     (03016) ;              *** SYNCHRONIZATION ***
                     (03017) ;
                     (03018) ;
                     (03019) ; 1) SHIFT MAX CORR. INDEX WORDS BACK ONE
                     (03020) ;          I.E. P(I-1)=P(I)
                     (03021) ;
      0519F 0000     (03022)          EVEN
      051A0 9010E009 (03023) SYNC0    MOVIR   R1,NOSMAX
      051A2 2711     (03024)          DECR    R1,1            R1, INDEX P(I)
      051A3 9020E009 (03025)          MOVIR   R2,NOSMAX       R2, INDEX P(I-1)
      051A5 F042E4BB (03026) SC01A    MOVMR   R4,PSMAX(R1)    MOV P(I)
      051A7 E044E4BB (03027)          MOVRM   R4,PSMAX(R2)            TO P(I-1)
      051A9 2721     (03028)          DECR    R2,1
      051AA 8C1A51A5 (03029)          DJP     R1,SC01A
                     (03030) ;
                     (03031) ; 2) UPDATE SYNC HISTOGRAM FOR CURRENT FRAME
                     (03032) ;
                     (03033) ;--- SET UP FOR UPDATING
                     (03034) ;
      051AC 903000BC (03035)          MOVIR   R3,SBC          R3=188
      051AE FC30E4B8 (03036)          ADDMR   R3,PSBASE       CURRENT BASE IND + 188
      051B0 9A3003FF (03037)          ANDIR   R3,$03FF
      051B2 0D40     (03038)          CLR     R4              MAX CORR INDEX THIS BLOCK
      051B3 0D60     (03039)          CLR     R5              MAX CORR VALUE
      051B4 90200179 (03040)          MOVIR   R2,$179         HISTOGRAM INDEX R2=377
                     (03041) ;
      051B6 F010E4CF (03042)          MOVMR   R1,VSSYNC       SYNC BIT VALUE TO R1
```

```
PAGE   78:        MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980


      0510 9A1#####1 (#3043)          ANDIR   R1,S####1        MASK LSB
                     (#3044) ;
      051A 987#####1 (#3045) SCS2#    MOVIR   R7,S1            READY TO MOVE IN BIT 1
           ####51DD  (#3046) ICB1#=#L+1
      051C FA7654AC  (#3047)          ANDMR   R7,BASICB(R3)    MOVE LSB OF INPUT WORD
      051E 4472      (#3048)          XORRR   R7,R1            RESULT OF CORR. OF SYNC WITH INFO BIT
                     (#3049) ;                                ;SAME = #  , DIFFERENT = 1
      051F 8#2#51EE  (#3050)          JMP     SCS22,GTZ        JUMP FOR DIFFERENT
                     (#3051) ;
                     (#3052) ;--- FOR MATCH WITH SYNC BIT
                     (#3053) ;
           ####51E2  (#3054) OQB11=#L+1
      51E1 F#6454AE  (#3055)          MOVMR   R6,BASOQB(R2)    R6=SYNC COR VAL FOR CURRENT BIT
      51E3 2661      (#3056)          INCR    R6,1             ADD 1 TO IT
      51E4 426A      (#3057)          CMPRR   R6,R5            CMPR WITH MAX SO FAR (R6-R5)
      51E5 8#3#51E9  (#3058)          JMP     SCS21,LTZ        THIS VALUE < MAX SO FAR
      51E7 4#5C      (#3059)          MOVRR   R5,R6            IF MAX, UPDATE MAX VALUE
      51E8 4#44      (#3060)          MOVRR   R4,R2                    UPDATE MAX INDEX
           ####51EA  (#3061) OQB12=#L+1
      51E9 E#6454AE  (#3062) SCS21    MOVRM   R6,BASOQB(R2)    SAVE NEW CORR VALUE
      51EB 2721      (#3063)          DECR    R2,1             DECR HIST INDEX
      51EC 8###51F9  (#3064)          JMP     SCS24
                     (#3065) ;
                     (#3066) ;--- FOR NO MATCH WITH SYNC BIT
                     (#3067) ;
      51EE 2721      (#3068) SCS22    DECR    R2,1              DECR HIST INDEX
           ####51F#  (#3069) OQB13=#L+1
      51EF F#6454AE  (#3070)          MOVMR   R6,BASOQB(R2)     SYNC CORR VAL TO R6
      51F1 2661      (#3071)          INCR    R6,1
      51F2 426A      (#3072)          CMPRR   R6,R5             (R6-R5)
      51F3 8#3#51F7  (#3073)          JMP     SCS23,LTZ          THIS VAL NOT MAX
      51F5 4#5C      (#3074)          MOVRR   R5,R6              IF MAX, UPDATE MAX VAL
      51F6 4#44      (#3075)          MOVRR   R4,R2                      UPDATE MAX INDEX
           ####51F8  (#3076) OQB14=#L+1
      51F7 E#6454AE  (#3077) SCS23    MOVRM   R6,BASOQB(R2)       SAVE NEW CORR VAL
                     (#3078) ;
                     (#3079) ;--- MERGE HERE BOTH CASES, MATCH OR NO-MATCH
                     (#3080) ;
      51F9 2731      (#3081) SCS24    DECR    R3,1                INP IND
      51FA 9A3###3FF (#3082)          ANDIR   R3,S##3FF           MOD 1#24
      51FC 8C2#51DA  (#3083)          DJP     R2,SCS2#            DEC HIST IND & LOOP
                     (#3084) ;
                     (#3085) ; ----  IF NOT SYNCD IN 1# SECONDS, REINITIALIZE
                     (#3086) ;
```

```
                    (03087) ;         CMPMR   R5,TSLIM        CMPR IN TERMS OF NO OF ITERATIONS
                    (03088) ;         JMP     SC$60,GTZ       >10 SECS, REINIT.
                    (03089) ;
                    (03090) ; 3) IF PRESENT MAX INDEX NOT THE SAME AS LAST TIME'S, CHECK IT OUT
                    (03091) ;
 051FE 8000052B8    (03092)           JMP     SC$31         ;NOTE: SKIP OVER THIS CHECK. TEMP!
                    (03093) ;
 05200 F020548B     (03094)           MOVMR   R2,PSMAX            PREV MAX IND.
 05202 4228         (03095)           CMPRR   R2,R4               (R2-R4)
 05203 8010052B8    (03096)           JMP     SC$31,EQZ           IF SAME JUMP
                    (03097) ;                                    ;IF NOT,
      00005206      (03098) OQB15=@L+1
 05205 F0648AE      (03099)           MOVMR   R6,BASOQB(R2)       GET OLD MAX VAL
 05207 426A         (03100)           CMPRR   R6,R5               (PAST VAL-PRESENT VAL)
 05208 8120052B8    (03101)           JMP     SC$31,LEZ            IF PAST VAL LESS,JMP
 0520A 4044         (03102)           MOVRR   R4,R2
 0520B E040548B     (03103) SC$31     MOVRM   R4,PSMAX             SAVE NEW MAX INDEX
                    (03104) ;
                    (03105) ; 4) CHECK IF SYNCHRONIZATION ACHIEVED
                    (03106) ;              IF ALL 10 PREV VALS OF MAX IND ARE THE SAME, WE HAVE SYNC
                    (03107) ;
 0520D 90200009     (03108)           MOVIR   R2,NOSMAX
 0520F F2405488     (03109) SC$41     CMPMR   R4,PSMAX(R2)        CMPR CURRENT & PREV VAL
 05211 8110052FD    (03110)           JMP     SC$70,NEZ           NOT EQUAL, NOT SYNCD
 05213 8C20052F     (03111)           DJP     R2,SC$41
                    (03112) ;
                    (03113) ; 5) SYNCHRONIZED. SET UP FOR RETURNING TO RECEIVER
                    (03114) ;
                    (03115) ;--- SET UP 2 BLOCKS SYNC VAL, (NOT NEXT)
                    (03116) ;
 05215 2A00         (03117)           SRBCL   0,R4                CHECK IND, EVEN OR ODD
 05216 8000521C     (03118)           JMP     SC$51               ODD-CURRENT SYNC BIT RIGHT,LEAVE
 05218 9070000001   (03119)           MOVIR   R7,S0000001         EVEN-CURRENT SYNC BIT IS WRONG
 0521A F570054CF    (03120)           XORRM   R7,VSSYNC           INVERT FOR 2 BLOCKS AWAY
 0521C 2000         (03121) SC$51     NOP     1
                    (03122) ;
                    (03123) ;--- UPDATE BASE OF CURRENT FRAME POINTER
                    (03124) ;
 0521D F030548B     (03125)           MOVMR   R3,PSBASE
 0521F 9C30000BD    (03126)           ADDIR   R3,SBD              BASE ADDR+189
 05221 9A3003FF     (03127)           ANDIR   R3,S03FF            MOD 1024
 05223 E030548B     (03128)           MOVRM   R3,PSBASE
                    (03129) ;
                    (03130) ;--- SET UP BIT-1 POINTER
```

```
                    (#3131) ;
#5225 3C41          (#3132)          LRS     R4,1                R4/2+BASE IS BIT-1 POINTER
#5226 4C46          (#3133)          ADDRR   R4,R3
#5227 9A4##3FF      (#3134)          ANDIR   R4,S##3FF           MOD 1#24
#5229 E#4#5489      (#3135)          MOVRM   R4,P$BIT1
                    (#3136) ;
                    (#3137) ;--- FUNCTION LIST SELECTOR UPDATE
                    (#3138) ;
#522B F#2#5486      (#3139)          MOVMR   R2,SID$ISL
#522D 9#7####1      (#314#)          MOVIR   R7,$1               ISL1 = 1, FOR DECODER
#522F E#74#5#2      (#3141)          MOVRM   R7,ISVT$(R2)        SET ISL1 TO +1
#5231 2621          (#3142)          INCR    R2,1                SID OF ISL2
#5232 F#64#5#2      (#3143)          MOVMR   R6,ISVT$(R2)        VAL OF ISL2
#5234 #B6#          (#3144)          NEG     R6
#5235 E#64#5#2      (#3145)          MOVRM   R6,ISVT$(R2)        ISL2=-ISL2
                    (#3146) ;
                    (#3147) ; 5) INITIALIZE BETA, T, AND Q-OUT FOR NEXT RECEIVER OPERATION
                    (#3148) ;
                    (#3149) ;--- INITIALIZE T=# I.E.2#, AND BETA=48 I.E. #.#
                    (#315#) ;
#5237 F#1#54B4      (#3151)          MOVMR   R1,BA$FCB           GET FCB POINTER
#5239 2612          (#3152)          INCR    R1,2                POINT TO ITB IN FCB
#523A 6#22          (#3153)          MOVWR   R2,R1               GET VALS OF ITB1 AND ITB2
#523B #26#          (#3154)          TEST    R6                  CHECK FUNCTION LIST # FOR NEXT OP.
#523C 183#          (#3155)          SKP     LTZ                 ISL2=-1, FUNC. LIST 1
#523D 3C28          (#3156)          LRS     R2,8                ISL2=1 , FUNC. LIST 2
#523E 9A2###FF      (#3157)          ANDIR   R2,$FF              KEEP APPROP. ITB
                    (#3158) ;
#524# #D1#          (#3159)          CLR     R1                  OUT T=#  I.E. 2#
#5241 3A21          (#316#)          LLS     R2,1
#5242 E#14#382      (#3161)          MOVRM   R1,SVT$(R2)
                    (#3162) ;
#5244 9#1###3#      (#3163)          MOVIR   R1,$3#              OUT BETA=48  I.E. #.#
#5246 2622          (#3164)          INCR    R2,2
#5247 E#14#382      (#3165)          MOVRM   R1,SVT$(R2)
                    (#3166) ;
                    (#3167) ;--- INITIALIZE NEXT Q-OUT BUFF [1 (NO PH BETA), 126  #'S, -12]
                    (#3168) ;
#5249 F#1#54B4      (#3169)          MOVMR   R1,BA$FCB
#524B 2611          (#317#)          INCR    R1,1
#524C 6#22          (#3171)          MOVWR   R2,R1               BID WORD
#524D #26#          (#3172)          TEST    R6
#524E 183#          (#3173)          SKP     LTZ                 ISL2=-1  FUNC LIST 1
#524F 3C28          (#3174)          LRS     R2,8                ISL2= 1  FUNC LIST 2
```

```
      0525 9A2000FF (0175)          ANDIR   R2,$FF
      0527 3A21     (0176)          LLS     R2,1
      0529 C044B582 (0177)          MOVMRL  R4,BCT$BA(R2)       ADDR OF NEXT Q-OUT BUFF
                    (0178) ;
      052D 90400014 (0179)          MOVIR   R4,$14              BUS 1, IND REG R2
      052F 8440B4AE (0180)          MOVRML  R4,BA$OQB           SAVE
                    (0181) ;
                    (0182) ;--- UPDATE PROGRAM LOC IN NEXT SECTION REFERRING TO OQB
                    (0183) ;
      0533 E005B262 (0184)          MOVRM   R5,OQB01
      0537 E005B26A (0185)          MOVRM   R5,OQB02
                    (0186) ;
                    (0187) ;
      053B 90200070 (0188)          MOVIR   R2,$7E
      053F 90600002 (0189)          MOVIR   R6,$2               Q-LEVEL 1*2
           0000B282 (0190) OQB01=@L+1
      0543 E0B454AE (0191) SC$52    MOVRM   R6,BA$OQB(R2)       126  1'S TO OQB
      0547 8C20B261 (0192)          DJP     R2,SC$52
                    (0193) ;
      054B 90200071 (0194)          MOVIR   R2,$7F
      054F 9060FFF4 (0195)          MOVIR   R6,$FFF4            END CODE= -12
           0000B26A (0196) OQB02=@L+1
      0553 E0B454AE (0197)          MOVRM   R6,BA$OQB(R2)
                    (0198) ;
                    (0199) ;--- INITIALIZE POINTER IN SHAT CIRCULAR BUFFER
                    (0200) ;
      0557 F060B4D6 (0201)          MOVMR   R6,SZ$SHAT
      055B 266F     (0202)          INCR    R6,15
      055C E060B4B7 (0203)          MOVRM   R6,P$SHAT
                    (0204) ;
                    (0205) ;--- SET UP BASE ADDR OF BETA DECODE TABLE, IND REG R7
                    (0206) ;
      0560 F010B4B4 (0207)          MOVMR   R1,BA$FCB
      0564 2613     (0208)          INCR    R1,3
      0566 7022B0FF (0209)          MOVWK   R2,R1,$00FF
      056A 3A21     (0210)          LLS     R2,1
      056C C044B582 (0211)          MOVMRL  R4,BCT$BA(R2)       R4,R5=BASE ADDR WORDS
      0570 9040001E (0212)          MOVIR   R4,$001E            BUS1, IND REG R7
      0574 8440B4B2 (0213)          MOVRML  R4,BA$DECB
                    (0214) ;
                    (0215) ;--- MODIFY PROGRAM LOCATIONS WHICH REFER TO DECB
                    (0216) ;
      0578 E005B2E6 (0217)          MOVRM   R5,DECB10
                    (0218) ;
```

```
0527E E050053DF (03219)          MOVRM   R5,DECB20
                (03220) ;
                (03221) ;
                (03222) ;--- SET UP BASE ADDR OF Q-OUT DECODE TABLE
                (03223) ;
05280 6022      (03224)          MOVWR   R2,R1
05281 3C27      (03225)          LRS     R2,7                    BID*2
05282 C044A582  (03226)          MOVMRL  R4,BCTSBA(R2)           R4,R5=BASE ADDR WORDS
05284 9040001E  (03227)          MOVIR   R4,$001E                BUS1, IND REG R7
05286 8440548A  (03228)          MOVRML  R4,BASDECT
                (03229) ;
                (03230) ;--- MODIFY PROGRAM LOCATIONS WHICH REFER TO DECT
                (03231) ;
05288 E050547A  (03232)          MOVRM   R5,DECT20
0528A E050543C  (03233)          MOVRM   R5,DECT21
                (03234) ;
                (03235) ;
                (03236) ;--- PARITY CHECK ON FIRST H-BLOCK
                (03237) ;
0528C F030548B  (03238)          MOVMR   R3,PSBIT1
0528E 9C300039  (03239)          ADDIR   R3,$39                  POINT TO FIRST PARITY BIT
05290 9A300FFF  (03240)          ANDIR   R3,$03FF                MOD 1024
05292 90500005  (03241)          MOVIR   R5,$0005                COUNTER FOR 6 BITS
05294 0D70      (03242)          CLR     R7
                (03243) ;
05295 3A71      (03244) SC053    LLS     R7,1
     00005297   (03245) ICB11=@L+1
05296 F0665AC   (03246)          MOVMR   R6,BASICB(R3)           GET INP WORD TO R6
05298 2631      (03247)          INCR    R3,1                    INP INDEX
05299 9A300FFF  (03248)          ANDIR   R3,$03FF                MOD 1024
0529B 567C0001  (03249)          IORKR   R7,R6,$0001             MOVE BIT TO R7
0529D 8C505295  (03250)          DJP     R5,SC053
                (03251) ;
0529F 0D60      (03252)          CLR     R6                      SET PARITY WORD TO 0
052A0 90400039  (03253)          MOVIR   R4,$0039
052A2 0B40      (03254)          NEG     R4                      COUNTER   -57 TO 0
052A3 F0305489  (03255)          MOVMR   R3,PSBIT1               BIT1 IND
                (03256) ;
052A5 2661      (03257) SC054    INCR    R6,1                    INCR PARITY WORD
052A6 90100001  (03258)          MOVIR   R1,$0001
     000052A9   (03259) ICB12=@L+1
052A8 FA1654AC  (03260)          ANDMR   R1,BASICB(R3)           MOV LSB TO R1
052AA 1810      (03261)          SKP     EQZ
052AB 447C      (03262)          XORRR   R7,R6                   UPDATE PARITY FOR 1
```

```
#52AC 2631      (#3263)          INCR    R3,1
#52AD 9A3##3FF  (#3264)          ANDIR   R3,S##3FF              MOD 1#24
                (#3265) ;
#52AF 894#52A5  (#3266)          IJN     R4,SC$54
                (#3267) ;
                (#3268) ;--- CHECK TO SEE IF ANY ERRORS
#52B1 #27#      (#3269)          TEST    R7                     TEST PARITY SYNDROME
#52B2 8#1#52BE  (#327#)          JMP     SC$541,EQZ             =#, NO ERROR, JUMP
#52B4 2771      (#3271)          DECR    R7,1
#52B5 F#3#54B9  (#3272)          MOVMR   R3,PSBIT1               SET UP INDEX FOR ERROR
#52B7 4C3E      (#3273)          ADDRR   R3,R7                   BIT1 IND + SYNDROME - 1
#52B8 9A3##3FF  (#3274)          ANDIR   R3,S##3FF               MOD 1#24
#52BA 9#5####1  (#3275)          MOVIR   R5,S###1
#52BC F5D#54AC  (#3276)          XORRM   R5,@BA$ICB              INVERT LSB AT Err LOC
                (#3277) ;
                (#3278) ;--- DECODE VALUE OF NEXT T (INP BUFF MUST CONTAIN MSB FIRST)
                (#3279) ;
#52BE F#3#54B9  (#328#) SC$541   MOVMR   R3,PSBIT1
#52C# 2631      (#3281)          INCR    R3,1
#52C1 9A3##3FF  (#3282)          ANDIR   R3,S##3FF
#52C3 9#4####5  (#3283)          MOVIR   R4,S###5
#52C5 #D7#      (#3284)          CLR     R7
                (#3285) ;
#52C6 3A71      (#3286) SC$55    LLS     R7,1
      ####52C8  (#3287) ICB13=#L+1
#52C7 F#1654AC  (#3288)          MOVMR   R1,BA$ICB(R3)
#52C9 5672###1  (#3289)          IORKR   R7,R1,S###1
#52CB 2631      (#329#)          INCR    R3,1
#52CC 9A3##3FF  (#3291)          ANDIR   R3,S##3FF
                (#3292) ;
#52CE 8C4#52C6  (#3293)          DJP     R4,SC$55
                (#3294) ;
#52D# E#7#54D#  (#3295)          MOVRM   R7,VST          SAVE VALUE OF T
                (#3296) ;
                (#3297) ;--- VALUE OF NEXT BETA, PHONEY BETA
                (#3298) ;
#52D2 9#7####1  (#3299)          MOVIR   R7,S1           TO INITIALIZE PHONEY BETA
#52D4 E#7#54D2  (#33##)          MOVRM   R7,VSPHB        INITIALIZE PHONEY BETA
#52D6 9#2####1  (#33#1)          MOVIR   R2,S1           LOOP COUNTER FOR 2
                (#33#2) ;
#52D8 #D7#      (#33#3) SC$56    CLR     R7              INDEX FOR BETA CODE
#52D9 9#4####6  (#33#4)          MOVIR   R4,S6           BETA CODE LENGTH = 7 BITS
                (#33#5) ;
#52DB 3A71      (#33#6) SC$57    LLS     R7,1
```

```
PAGE   84:        MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980


      0000052DD (00307) ICB14=@L+1
0052DC F0165AAC (00308)           MOVMR   R1,BASICB(R3)   ;COLLECT BETA INDEX
0052DE 5672000A1 (00309)          IORKR   R7,R1,$1        ;
                (00310) ;
0052E0 2631     (00311)           INCR    R3,1            ;
0052E1 9A3000FF (00312)           ANDIR   R3,$00FF        ; INP INDEX
                (00313) ;
0052E3 0C40052D8 (00314)          DJP     R4,SC$57
                (00315) ;
      0000052E6 (00316) DECB10=@L+1
0052E5 F006E5AB2 (00317)          MOVMR   R6,BA$DECB(R7)  CODE FOR BETA
0052E7 0130052EF (00318)          JMP     SC$58,GEZ       TRUE BETA
0052E9 E006054D2 (00319)          MOVRM   R6,VSPHB        SAVE PHONEY BETA
                (00320) ;
0052EB 0C20052D8 (00321)          DJP     R2,SC$56
                (00322) ;
                (00323) ;                                 ;ERROR. GOT 2 PHONEY BETAS
                (00324) ;                                 ;FORCE BETA TO 0
                (00325) ;
0052ED 9060003B (00326)           MOVIR   R6,$003B        BETA LEVEL 48 <--> 0.0
0052EF E006054D1 (00327) SC$58    MOVRM   R6,VSBETA       SAVE BETA VALUE
                (00328) ;
                (00329) ;--- INITIALIZE NEXT-INFO-BIT POINTER
                (00330) ;
0052F1 E003054BA (00331)          MOVRM   R3,PSINFO
                (00332) ;
                (00333) ;--- INIT SYNC VARIABLE TO -1
                (00334) ;
0052F3 0C60     (00335)           CCR     R6
0052F4 E006054D3 (00336)          MOVRM   R6,VSPAR
                (00337) ;
                (00338) ;--- ALL SET UP FOR DECODER. GO FOR XMTR UPDATE
                (00339) ;
0052F6 0000054A6 (00340)          JMP     DC$90           CALL FOR XMTR UPD
                (00341) ;
                (00342) ;
                (00343) ; 6) IF NOT SYNCD IN 10 SECONDS, RE-INITIALIZE
                (00344) ;
0052F8 F003054B6 (00345) SC$60    MOVMR   R3,SID$ISL      GET SID # FOR ISL 1
0052FA 0D60     (00346)           CLR     R6
0052FB E006605A2 (00347)          MOVRM   R6,ISVT$(R3)    ISL1=0, REINIT.
                (00348) ;
                (00349) ; 7) NOT YET SYNCD. SET UP FOR ONE MORE SYNC OPERATION
                (00350) ;
```

```
                    (03351) ;--- INVERT SYNC VALUE
                    (03352) ;
     0052FD 9001FFFF1 (03353) SC$70    MOVIR   R1,$FFF1
0054CF (03354)       XORRM   R1,VSSYNC                                          052FF F51
                    (03355) ;
                    (03356) ;--- UPDATE BASE OF CURRENT BLOCK FRAME
                    (03357) ;
     005301 F03054B8 (03358)           MOVMR   R3,PSBASE
     005303 9C3000BD (03359)           ADDIR   R3,SBD                  BASE IND + 189
     005305 9A3003FF (03360)           ANDIR   R3,$03FF                MOD 1024
     005307 E0305488 (03361)           MOVRM   R3,PSBASE
                    (03362) ;
                    (03363) ;--- UPDATE FUNCTION LIST SELECTOR ISL2
                    (03364) ;
     005309 F0305486 (03365)           MOVMR   R3,SID$ISL
     00530B 2631     (03366)           INCR    R3,1
     00530C F0665052 (03367)           MOVMR   R6,ISVT$(R3)
     00530E 0B60     (03368)           NEG     R6
     00530F E0665052 (03369)           MOVRM   R6,ISVT$(R3)
                    (03370) ;
                    (03371) ;--- READY TO RETURN. GO FOR XMTR UPDATE
                    (03372) ;
     005311 00005406 (03373)           JMP     DC$90             CALL FOR XMTR UPD
                    (03374) ;
                    (03375) ;
                    (03376) ;
                    (03377) ;                    *** DECODER ***
                    (03378) ;
                    (03379) ;
                    (03380) ; 1) GET ISL2 TO DECIDE FUNC LIST 1 OR 2
                    (03381) ;
     005313 0000     (03382)           EVEN
     005314 F0105484 (03383) DECDS     MOVMR   R1,BASFCB
     005316 F0205486 (03384)           MOVMR   R2,SID$ISL
     005318 2621     (03385)           INCR    R2,1                     SID FOR ISL2
     005319 F0345052 (03386)           MOVMR   R3,ISVT$(R2)             ISL2
                    (03387) ;
     00531B 2001     (03388)           NOP     2                        FOR TIMING INSTRUCTION
     00531C 2000
                    (03389) ;
                    (03390) ; 2) BASE ADDR FOR Q-OUT AND SID FOR T,BETA
                    (03391) ;
     00531D 0030533D (03392)           JMP     DC$20,LTZ                 ISL=-1,FUNC LIST1,BUFF#2
                    (03393) ;                                           ;ISL= 1,FUNC LIST2,BUFF#1
```

```
#531F 2611     (#3394)          INCR    R1,1
#5320 7#22##FF (#3395)          MOVWK   R2,R1,S##FF             BID FOR OQB1
#5322 3A21     (#3396)          LLS     R2,1
#5323 C#44#582 (#3397)          MOVMRL  R4,BCT$BA(R2)           R4,R5=BASE ADDR
#5325 9#4###12 (#3398)          MOVIR   R4,S##12                BUS1,IND REG R1
#5327 844#54AE (#3399)          MOVRML  R4,BA$OQB               SAVE
               (#34##) ;
#5329 2611     (#34#1)          INCR    R1,1
#532A 7#22##FF (#34#2)          MOVWK   R2,R1,S##FF
#532C E#2#54B5 (#34#3)          MOVRM   R2,SID$TB               SAVE SID FOR T,BETA
#532E 8###533E (#34#4)          JMP     DC$21
               (#34#5) ;
               (#34#6) ;
#5330 2611     (#34#7) DC$2#    INCR    R1,1                   ISL=-1  BUFF#2
#5331 6#22     (#34#8)          MOVWR   R2,R1
#5332 3C27     (#34#9)          LRS     R2,7
#5333 C#44#582 (#341#)          MOVMRL  R4,BCT$BA(R2)           R4,R5=BASE ADDR
#5335 9#4###12 (#3411)          MOVIR   R4,S##12                BUS 1, IND REG R1
#5337 844#54AE (#3412)          MOVRML  R4,BA$OQB
               (#3413) ,
#5339 2611     (#3414)          INCR    R1,1
#533A 6#22     (#3415)          MOVWR   R2,R1
#533B 3C28     (#3416)          LRS     R2,8
#533C E#2#54B5 (#3417)          MOVRM   R2,SID$TB               SAVE SID FOR T,BETA
               (#3418) ;
               (#3419) ;--- MODIFY PROGRAM LOCATIONS WHICH REFER TO OQB
               (#342#) ;
#533E 2###     (#3421) DC$21    NOP                            R5 CONTAINS OQB ADDR
#533F E#5#547C (#3422)          MOVRM   R5,OQB26
#5341 E#5#5487 (#3423)          MOVRM   R5,OQB27
               (#3424) ;
               (#3425) ;
               (#3426) ; 3) PARITY CHECK OVER 3 HAMMING BLOCKS
               (#3427) ;
#5343 9#2####2 (#3428)          MOVIR   R2,S###2            COUNTER FOR 3 BLOCKS
#5345 F#5#54B9 (#3429)          MOVMR   R5,P$BIT1           SAVE BIT1 POINTER IN R5
               (#343#) ;
#5347 9C5###3F (#3431) DC$3#    ADDIR   R5,S##3F            POINT TO NEXT FRAME I.E.+63
#5349 9A5##3FF (#3432)          ANDIR   R5,S#3FF
               (#3433) ;
#534B 9#3###39 (#3434)          MOVIR   R3,S##39
#534D 4C3A     (#3435)          ADDRR   R3,R5               POINT TO PARITY BIT
#534E 9A3##3FF (#3436)          ANDIR   R3,S#3FF
               (#3437) ;
```

```
0536 0D70     (0340)          CLR     R7
0537 9040005  (0339)          MOVIR   R4,$0005            COUNTER FOR 6 BITS
0539 3A71     (0344) DC031    LLS     R7,1
     0000535B (0341) ICB20=@L+1
053A F0654AC (0342)          MOVMR   R6,BASICB(R3)       NEXT PARITY BIT
053C 567C001 (0343)          IORKR   R7,R6,$0001         LOAD TO LSB OF PARITY WORD
             (0344) ;
053E 2631     (0345)          INCR    R3,1
053F 9A300FF  (0346)          ANDIR   R3,$03FF            INP INDEX
             (0347) ;
0541 8C40539  (0348)          DJP     R4,DC031
             (0349) ;
             (0350) ;--- PARITY CHECK ON 57 BITS
             (0351) ;
0544 403A     (0352)          MOVRR   R3,R5               SET UP INP INDEX
0545 9040038  (0353)          MOVIR   R4,$0038            COUNTER 56 - 0
0548 0D60     (0354)          CLR     R6
             (0355) ;
0549 2661     (0356) DC032    INCR    R6,1
     0000054B (0357) ICB21=@L+1
054A F01654AC (0358)          MOVMR   R1,BASICB(R3)       INFO WORD
054C 2802     (0359)          SRBC    0,R1                MASK BIT 1
054D 447C     (0360)          XORRR   R7,R6               PARITY UPDATE FOR 1
             (0361) ;
054F 2631     (0362)          INCR    R3,1                INP IND
0550 9A300FF  (0363)          ANDIR   R3,$03FF
             (0364) ;
0552 8C40549  (0365)          DJP     R4,DC032
             (0366) ;
             (0367) ;--- CORRECT ANY ERROR
             (0368) ;
0555 0270     (0369)          TEST    R7
0556 9010560  (0370)          JMP     DC033,EQZ
0559 2771     (0371)          DECR    R7,1
055A 403A     (0372)          MOVRR   R3,R5
055C 4C3E     (0373)          ADDRR   R3,R7               SET UP ERROR IND
055D 9A300FF  (0374)          ANDIR   R3,$03FF
055F 9010001  (0375)          MOVIR   R1,$0001
     0000560 (0376) ICB22=@L+1
0560 F51654AC (0377)          XORRM   R1,BASICB(R3)
             (0378) ;
0563 8C20543  (0379) DC033    DJP     R2,DC030            LOOP FOR 3 H-FRAMES
             (0380) ;
             (0381) ; 4) UPDATE FOR SYNC MONITOR
```

```
                    (0348Z) ;           V(I+1)=RIGHT SHIFT 1 [V(I)], IF CORRELATED
                    (0348Z) ;           V(I+1)=LEFT  SHIFT 2 [V(I)], NOT CORRELATED
                    (0348Z) ;
                    (0348Z) ;           BIT 11 CLEAR ==> SYNC LOST
                    (0348Z) ;
0379 403A           (0348Z)             MOVRR    R3,R5      POINT TO NEXT SYNC BIT
                    (0348Z) ;
037A F07054D3       (0348Z)             MOVMR    R7,VSPAR   GET SYNC VARIABLE
                    (0349Z) ;
037C 901000F1       (0349Z)             MOVIR    R1,S000F1     CHECK CORR. IN THIS FRAME
037E F4105CF        (0349Z)             XORMR    R1,VSSYNC
0380 9A1000F1       (0349Z)             ANDIR    R1,S000F1
            00005383(0349Z) ICB23=@L+1
0382 F41654AC       (0349Z)             XORMR    R1,BASICB(R3)  1 - MATCH , 0 - NO MATCH
0384 2A02           (0349Z)             SRBCL    0,R1
0385 3871           (0349Z)             ARS      R7,1          CORRELATED
0386 1800           (0349Z)             SKP      0
0387 3A72           (0349Z)             LLS      R7,2          UNCORRELATED
                    (0350Z) ;
0388 E07054D3       (0351Z)             MOVRM    R7,VSPAR      SAVE NEW VALUE
038A 2B3E           (0352Z)             SRBCL    11,R7         IF BIT 11 CLEAR, SYNC LOST
038B 8000539A2      (0353Z)             JMP      DC$50         SYNC RETAINED, GO TO DECODE
                    (0354Z) ;
                    (0356Z) ;--- IF SYNC LOST, SET UP FOR SYNC ACQUISITION NEXT TIME
                    (0356Z) ;
038D F07054B8       (0357Z)             MOVMR    R7,PSBASE
038F 9C7000BD       (0358Z)             ADDIR    R7,SBD        UPDATE BASE OF FRAME POINTER
0391 9A7003FF       (0359Z)             ANDIR    R7,S03FF
0393 E07054B8       (0351Z)             MOVRM    R7,PSBASE
                    (0351Z) ;
                    (0352Z) ;--- UPDATE FUNCTION SELECT FLAGS AND RETURN
                    (0353Z) ;                    ISL1=0      ISL2=-ISL2
                    (0354Z) ;
0395 F07054B6       (0355Z)             MOVMR    R7,SIDSISL
0397 0D60           (0356Z)             CLR      R6
0398 E06E05Z2       (0357Z)             MOVRM    R6,ISVTS(R7)
039A 2671           (0358Z)             INCR     R7,1
039B F06E05Z2       (0359Z)             MOVMR    R6,ISVTS(R7)
039D 0860           (0352Z)             NEG      R6
039E E06E05Z2       (0361Z)             MOVRM    R6,ISVTS(R7)
                    (0362Z) ;
                    (0363Z) ;--- GO FOR XMTR UPDATE
                    (0364Z) ;
03A0 8000054A6      (0365Z)             JMP      DC$90                    JUMP FOR XMTR UPD
```

```
                    (0352B) ;
                    (0352C) ;
                    (0352D) ; 5) MOV T,BETA,PH BETA TO THEIR LOCATIONS
                    (0352E) ;
                    (0352F) ;--- MOVE T TO INTEGER SCALAR TABLE
                    (03530) ;
0053A2 F807054B5    (03531) DC.50    MOVMR   R7,SIDSTB
0053A4 3A71         (03532)          LLS     R7,1
0053A5 F806054D0    (03533)          MOVMR   R6,VST
0053A7 E806E0302    (03534)          MOVRM   R6,SVTS(R7)
                    (03535) ;
                    (03536) ;--- MOVE BETA TO ITS LOCATION IN INTEGER SCALAR TABLE
                    (03537) ;
0053A9 2672         (03538)          INCR    R7,2
0053AA F806054D1    (03539)          MOVMR   R6,VSBETA
0053AC E806E0302    (03540)          MOVRM   R6,SVTS(R7)
                    (03541) ;
                    (03542) ;--- NOW MOVE PHONEY BETA TO LOCATION IN OUTPUT BUFFER
                    (03543) ;
0053AE F801054AF    (03544)          MOVMR   R1,BASOQB+1          INITIALIZE OUTPUT POINTER
0053B0 F806054D2    (03545)          MOVMR   R6,VSPHB             GET PHONEY BETA VALUE
                    (03546) ;
                    (03547) ;--- UPDATE SHAT BUFFER POINTER IF PHONEY BETA
                    (03548) ;
0053B2 8020053B8    (03549)          JMP     DC$51,GTZ            NO PH. BETA, JUMP AND PROCEED
0053B4 F807054D5    (03550)          MOVMR   R7,VSGAP             GET GAP SIZE
0053B6 FD705487     (03551)          ADDRM   R7,P$SHAT            UPDATE SHAT POINTER
                    (03552) ;
0053B8 341C         (03553) DC$51    PUSHXI  R1,R6                OUTPUT PH. BETA VALUE
                    (03554) ;
                    (03555) ; 6) DECODE NEXT T,BETA, PH BETA  (R3,R5 STILL HAV LOC FOR NEXT FRAME)
                    (03556) ;
                    (03557) ;--- FIRST T
                    (03558) ;
0053B9 2631         (03559)          INCR    R3,1
0053BA 9A30003FF    (03560)          ANDIR   R3,$03FF              POINT TO 1ST BIT (MSB) OF T
                    (03561) ;
0053BC 9040000005   (03562)          MOVIR   R4,$5
0053BE 0D70         (03563)          CLR     R7
                    (03564) ;
0053BF 3A71         (03565) DC$60    LLS     R7,1
       000053C1     (03566) ICB24=@L+1
0053C0 F806054AC    (03567)          MOVMR   R6,BASICB(R3)
0053C2 667C0001     (03568)          IORKR   R7,R6,$0001
```

```
                    (0357C) ;
 053C4 2631         (03571)          INCR    R3,1          INP IND
 053C5 9A3003FF     (03572)          ANDIR   R3,S03FF
                    (03573) ;
 053C7 0C4053BF     (03574)          DJP     R4,DC860
                    (03575) ;
 053C9 E707054D0    (03576)          MOVRM   R7,V8T         SAVENEW T
                    (03577) ;
                    (03578) ;--- NEXT CHECK BETA TWICE
                    (03579) ;
 053CB 9070000001   (03580)          MOVIR   R7,S1          TO INITIALIZE PHONEY BETA
 053CD E70754D2     (03581)          MOVRM   R7,VSPHB       INITIALIZE PHONEY BETA
 053CF 9020000001   (03582)          MOVIR   R2,S1
                    (03583) ;
 053D1 9040000006   (03584) DC861    MOVIR   R4,S6
 053D3 0D70         (03585)          CLR     R7
                    (03586) ;
 053D4 3A71         (03587) DC862    LLS     R7,1
      0000053D6     (03588) IC825=@L+1
 053D5 F0660540C    (03589)          MOVMR   R6,BASICB(R3)
 053D7 867C0001     (03590)          IORKR   R7,R6,S1
                    (03591) ;
 053D9 2631         (03592)          INCR    R3,1
 053DA 9A3003FF     (03593)          ANDIR   R3,S03FF
                    (03594) ;
 053DC 0C4053D4     (03595)          DJP     R4,DC862
                    (03596) ;
      0000053DF     (03597) DEC820=@L+1
 053DE F06E54B2     (03598)          MOVMR   R6,BASDECB(R7)        DECODE VAL OF BETA
                    (03599) ;
 053E0 813053E8     (03600)          JMP     DC863,GEZ
                    (03601) ;
 053E2 E0605402     (03602)          MOVRM   R6,VSPHB
                    (03603) ;
 053E4 0C2053D1     (03604)          DJP     R2,DC861
                    (03605) ;
 053E6 9060000030   (03606)          MOVIR   R6,S0030           THIS POSITION MEANS ERROR IN BETA
                    (03607) ;                                  ;FORCE BETA TO 0.0
 053E8 E0605401     (03608) DC863    MOVRM   R6,VSBETA
                    (03609) ;
                    (03610) ;--- HANG ONTO 1ST INFO BIT IN NEXT FRAME (IN R5)
 053EA 4056         (03611)          MOVRR   R5,R3
                    (03612) ;
                    (03613) ; 7) NOW TO DECODE QUANTIZER OUTPUTS
```

```
                    (03614) ;
                    (03615) ;          THREE LOOPS OF 57 BITS
                    (03616) ;
03E8 F030548A       (03617)          MOVMR   R3,PSINFO
03ED F020548B       (03618)          MOVMR   R2,PSBIT1
03EF 4E34           (03619)          SUBRR   R3,R2
03F0 9A30003FF      (03620)          ANDIR   R3,$03FF
03F2 90400038       (03621)          MOVIR   R4,$38
03F4 4E46           (03622)          SUBRR   R4,R3                  INITIALIZE LOOP COUNTER
                    (03623) ;
03F6 F030548A       (03624)          MOVMR   R3,PSINFO              INP INDEX
03F7 90200002       (03625)          MOVIR   R2,$2                  LOOP COUNTER
03F9 0D70           (03626)          CLR     R7                     CODE ACCUMULATOR
03FA 000053FE       (03627)          JMP     DC$71
                    (03628) ;
03FC 90400038       (03629) DC$70    MOVIR   R4,$38
                    (03630) ;
03FE 3A71           (03631) DC$71    LLS     R7,1
     0004548B       (03632) IC$26=@L+1
0403 F066548C       (03633)          MOVMR   R6,BASICB(R3)
0407 567C0001       (03634)          IORKR   R7,R6,$1
                    (03635) ;
0403 2631           (03636)          INCR    R3,1
0404 9A30003FF      (03637)          ANDIR   R3,$03FF
                    (03638) ;
     0004547         (03639) DECT2$=@L+1
0406 F06E5450       (03640)          MOVMR   R6,BASDECT(R7)
040B 0030541B       (03641)          JMP     DC$72,LTZ              <0  NULL OR RUN LENGTH
040A 0020542A       (03642)          JMP     DC$73,GTZ              >0  CODE WORD
                    (03643) ;                                       ; =0  NOT YET CODE WORD
040C 0C403FE        (03644)          DJP     R4,DC$71               LOOP BACK FOR NEXT INP
                    (03645) ;
040E 0000542B       (03646)          JMP     DC$75                  END OF H-FRAME PROCESSING
                    (03647) ;
                    (03648) ;--- NULL OR RUN LENGTH CODE
                    (03649) ;
0410 F26054D4       (03650) DC$72    CMPMR   R6,VSTHR
0412 0020545B       (03651)          JMP     DC$81,GTZ                +VE  NULL CODE
                    (03652) ;
0414 F0705115       (03653)          MOVMR   R7,LSRUN               COUNTER FOR 14 1'S
0416 90600002       (03654)          MOVIR   R6,$2                  GONNA OUTPUT LEVEL 1*2
0418 4C1E           (03655)          ADDRR   R1,R7                  SET UP OUT POINTER
                    (03656) ;
0419 1C70           (03657)          RPT     R7                     OUTPUT TO BUFFER
```

```
0541A E062####  (03650)          MOVRM   R6,#(R1)
                (03659) ;
0541C 0D70      (03660)          CLR     R7                  FOR NEXT CODE WORD
0541D FC1#0115  (03661)          ADDMR   R1,LSRUN            SET UP OUT POINTER AFTER
0541F 2612      (03662)          INCR    R1,2                RPT INSTR HAS DECR IT BY LRUN+2
                (03663) ;
05420 0C40E3FE  (03664)          DJP     R4,DC071            LOOP FOR NEXT INP
                (03665) ;
05422 0###5420  (03666)          JMP     DC075               END OF H-FRAME PROCESSING
                (03667) ;
                (03668) ;--- CODE WORD
                (03669) ;
05424 341C      (03670) DC073    PUSHXI  R1,R6
05425 0D70      (03671)          CLR     R7
                (03672) ;
05426 0C40E3FE  (03673) DC074    DJP     R4,DC071
                (03674) ;
                (03675) ;--- AT THE END OF EACH H-FRAME
                (03676) ;
05428 2636      (03677) DC075    INCR    R3,6
05429 0A30#3FF  (03678)          ANDIR   R3,0#3FF
                (03679) ;
0542B 0C2#E3FC  (03680) DC076    DJP     R2,DC07#            LOOP FOR 3 H-FRAMES
                (03681) ;
                (03682) ;--- END OF 3 H-FRAMES I.E. ONE INFO FRAME OF 189 BITS
                (03683) ;
0542D 4#3A      (03684)          MOVRR   R3,R5                ;AT THE END OF 189 FRAME
                (03685) ;                                    ;UPDATE TO NEXT INFO BIT
                (03686) ;
0542E 0260      (03687)          TEST    R6
0542F 0#2#5464  (03688)          JMP     DC084,GTZ       IF FRAME COVERED EXACTLY QUIT
                (03689) ;
05431 904#####A (03690)          MOVIR   R4,0A           LAST LEVEL, 11 MORE BITS
                (03691) ;
05433 3A71      (03692) DC077    LLS     R7,1
      #####5435 (03693) IC827=#L+1
05434 F#6654AC  (03694)          MOVMR   R6,BASICB(R3)
05436 567C###1  (03695)          IORKR   R7,R6,0###1
                (03696) ;
05438 2631      (03697)          INCR    R3,1
05439 0A30#3FF  (03698)          ANDIR   R3,0#3FF
                (03699) ;
      ####543C  (037##) DECT21=#L+1
0543B F#6E540#  (037#1)          MOVMR   R6,BASDECT(R7)
```

```
0543D 8810544E (03702)          JMP     DC879,EQZ       NOT YET CODE
0543F 8820545Z (03703)          JMP     DC880,GTZ       CODE WORD
                (03704) ;                               ;LTZ, NULL OR RUN LENGTH
                (03705) ;--- NULL CODE
                (03706) ;
05441 F260540A (03707)          CMPMR   R6,VSTHR        THRESHOLD=-16
05443 8820546A (03708)          JMP     DC884,GTZ       >THR , NULL CODE
                (03709) ;                               ;<THR , RUN LENGTH
                (03710) ;
                (03711) ;--- RUN LENGTH CODE, OUTPUT 14 1'S
                (03712) ;
05445 F870B110 (03713)          MOVMR   R7,LSRUN            COUNTER FOR 14 1'S
05447 9860000Z (03714)          MOVIR   R6,02               LEVEL 1*2
                (03715) ;
05449 341C      (03716) DC878   PUSHXI  R1,R6               OUTPUT TO BUFFER
0544A 8C70B449 (03717)          DJP     R7,DC878
                (03718) ;
                (03719) ;
0544C 0000B464 (03720)          JMP     DC884
                (03721) ;
0544E 8C40B433 (03722) DC879    DJP     R4,DC877
0545Z 0000B464 (03723)          JMP     DC884
                (03724) ;
                (03725) ;--- LAST CODE WORD
                (03726) ;
05452 341C      (03727) DC880   PUSHXI  R1,R6
                (03728) ;
05453 0000B464 (03729)          JMP     DC884
                (03730) ;
05455 403A      (03731) DC881   MOVRR   R3,R5
                (03732) ;
                (03733) ;--- AFTER A NULL CODE, PUT IN 5 EXTRA LEVEL 1'S
                (03734) ;    ONLY IF # SAMPLES OUT = 126
                (03735) ;
05456 9860007F (03736) DC882   MOVIR   R6,07F          126+1
05458 4E62      (03737)         SUBRR   R6,R1           126-(ADDR+N)
05459 FC60B4AF (03738)          ADDMR   R6,BASOQB+1       126-(BASE ADDR+N)+BASE ADDR
0545B 8110B464 (03739)          JMP     DC884,NEZ       N .NE. 126, NO EXTRA SAMPLES
                (03740) ;
0545D 9870000A (03741)          MOVIR   R7,04           COUNTER FOR 5 1'S
0545F 9860000Z (03742)          MOVIR   R6,02            LEVEL 1*2
                (03743) ;
05461 341C      (03744) DC883   PUSHXI  R1,R6             OUTPUT LEVEL 1'S
05462 8C70B461 (03745)          DJP     R7,DC883
```

```
                    (0374) ;
                    (0374) ;
                    (0374) ;--- CHECK RECEIVER BUFFER FULL
                    (0374) ;
0546A F07054D6 (0375) DC$84     MOVMR   R7,SZ$SHAT      GET REC BUFFER SIZE
0546C F0605487 (0375) DC$84     MOVMR   R6,P$SHAT       REC BUFFER POINTER
0546E 9E60007F (0376) DC$84     SUBIR   R6,$7F          SAMPLES REMOVED FROM BUFF. 126+1
05470 FE105487 (0376) DC$84     SUBMR   R1,BA$OQB+1     COMPUTE # SAMPLES OUTPUT
05472 4C62     (0376) DC$84     ADDRR   R6,R1           SAMPLES ADDED TO SHAT BUFF.
05474 426E     (0376) DC$84     CMPRR   R6,R7           (BUFF. POINTER - BUFF. SIZE)
05476 8120547D (0376) DC$84     JMP     DC$86,LEZ       NOT FULL, JMP AND PROCEED
05478 4E6E     (0376) DC$84     SUBRR   R6,R7           NO. OF EXTRA SAMPLES
0547A 4E1C     (0376) DC$84     SUBRR   R1,R6           DELETE THE EXTRA SAMPLES
0547C 406E     (0376) DC$84     MOVRR   R6,R7           SHAT POINTER = 1024
0547E 80000481 (0377) DC$84     JMP     DC$89           GO FOR END CODE
                    (0378) ;
                    (0378) ;--- CHECK RECEIVER BUFFER EMPTY CONDITION
                    (0378) ;
05482 92600000 (0379) DC$86     CMPIR   R6,$0           (BUFF POINTER - 0)
05484 81305481 (0380) DC$86     JMP     DC$89,GEZ       NOT EMPTY, PROCEED
                    (0381) ;                            ;IF EMPTY, FILL WITH LEVEL 1'S
05486 90700002 (0382) DC$86     MOVIR   R7,$2           LEVEL 1*2
      00005484 (0383) OQB26=@L+1
0548A E0725AE (0384) DC$87     MOVRM   R7,BA$OQB(R1)   OUTPUT LEVEL 1
0548C 2611     (0385) DC$87     INCR    R1,1            OUT IND
0548E 8960548A (0386) DC$87     IJN     R6,DC$87
                    (0387) ;
05490 0D60     (0388) DC$87     CLR     R6              SHAT POINTER = 0
                    (0389) ;
05492 E0605487 (0390) DC$89     MOVRM   R6,P$SHAT       SAVE NEW SHAT POINTER
                    (0391) ;
                    (0392) ;--- NULL CODE
                    (0393) ;
05496 90600004 (0394) DC$89     MOVIR   R6,$C
05498 0960     (0395) DC$89     NEG     R6                  NULL CODE = -12
      00005497 (0396) OQB27=@L+1
05499 E0625AE (0397) DC$89     MOVRM   R6,BA$OQB(R1)
                    (0398) ;
                    (0399) ;
                    (0400) ; 0, UPDATE FOR NEXT OPERATION OF DECODER
                    (0401) ;
                    (0402) ;--- NEXT INFO BIT POINTER
0549D E0305489 (0403) DC$89     MOVRM   R3,P$INFO
                    (0404) ;
```

```
                    (0379D) ;--- NEXT BIT1 POINTER
                    (03791) ;
 054DA F805054B9 (03792)          MOVMR   R5,PSBIT1
 054DC 9C50008D (03793)           ADDIR   R5,SBD
 054DE 9A5003FF (03794)           ANDIR   R5,S03FF
 054E0 E805054B9 (03795)          MOVRM   R5,PSBIT1
                    (03796) ;
                    (03797) ;--- NEXT BASE OF FRAME
 05492 F803054B8 (03798)          MOVMR   R3,PSBASE
 05494 9C30008D (03799)           ADDIR   R3,SBD
 05496 9A3003FF (03800)           ANDIR   R3,S03FF
 05498 E803054B8 (03801)          MOVRM   R3,PSBASE
                    (03802) ;
                    (03803) ;--- NEXT SYNC VALUE
 0549A 9807FFFF1 (03804)          MOVIR   R7,SFFF1
 0549C F57054CF (03805)           XORRM   R7,VSSYNC
                    (03806) ;
                    (03807) ;--- FUNCTION LIST SELECTOR ISL2
 0549E F802054B6 (03808)          MOVMR   R2,SIDSISL
 054A0 2621     (03809)           INCR    R2,1
 054A1 F8340502 (03810)           MOVMR   R3,ISVTS(R2)
 054A3 0830     (03811)           NEG     R3
 054A4 E8340502 (03812)           MOVRM   R3,ISVTS(R2)
                    (03813) ;
                    (03814) ;
                    (03815) ;--- ALL DONE. SO RETURN
                    (03816) ;
                    (03817) ;--- UPDATE XMTR BEFORE RETURNING
                    (03818) ;
 054A6 0E70     (03819) DC$90     RETURN
                    (03820) ;DC$90    MOVMR   R1,SIDSISL      SID FOR FUNC SEL.
 054A7 2612     (03821)           INCR    R1,2            POINT TO IFSEL3
 054A8 9C100502 (03822)           ADDIR   R1,ISVTS        COMPUT ADDR FOR IFSEL3
 054AA 0000E836 (03823)           JMP     TXS             GO TO XMTR UPDATE MODULE
                    (03824) ;
                    (03825) ;--- LEAVE SPACE FOR VARIOUS VARIABLES
                    (03826) ;
                    (03827)           EVEN
 054AC 00000000 (03828) BASICB    DATA    F'0.0'
 054AE 00000000 (03829) BASOQB    DATA    F'0.0'
 054B0 00000000 (03830) BASDECT   DATA    F'0.0'
 054B2 00000000 (03831) BASDECB   DATA    F'0.0'
 054B4 0000     (03832) BASFCB    DATA    S0
 054B5 0000     (03833) SIDSTB    DATA    S0
```

```
PAGE   96:        MAP MODULES FOR THE  P A R C  ALGORITHM    --- JAN. 30, 1980

      0054B6 0000     (00834) SID$ISL   DATA     $0
      0054B7 0000     (00835) PS$HAT    DATA     $0
      0054B8 0000     (00836) PSBASE    DATA     $0000
      0054B9 0000     (00837) PSBITI    DATA     $0
      0054BA 0000     (00838) PSINFO    DATA     $0
      0054BB 00000000 (00839) PSMAX     DATA     10F'0.0'
       ...
      0054CF 0001     (00840) VS$YNC    DATA     $1
      0054D0 0000     (00841) VST       DATA     $0
      0054D1 0000     (00842) VSBETA    DATA     $0
      0054D2 0000     (00843) VSPHB     DATA     $0
      0054D3 4000     (00844) VSPAR     DATA     $4000
      0054D4 FFE0     (00845) VSTHR     DATA     $FFE0
      0054D5 0050     (00846) VSGAP     DATA     $50
      0054D6 0380     (00847) SZ$SHAT   DATA     $380             ;SIZE=944
      0054D7 0050     (00848) TSLIM     DATA     $50
                      (00849) *LSRUN     DATA     $D
                      (00850) ;
                      (00851) ;
                      (00852) ;--- UPDATE TOP OF MEMORY
                      (00853) ;
           00054D8    (00854)           TOESNEW=*L
           00000288   (00855)           *L=TOESPRT
      0028 0001054D8  (00856)           ADDR    TOESNEW(,BUS1$)
      028A            (00857)           END
```

```
ADSBETA:   #518C (#2481) (#2489) (#2654)
ADSENCT:   #518A (#2393) (#2491) (#2521) (#2571) (#2653)
ADSIN:     #515A (#2795) (#2815)
ADSOLDB:   #518E (#2476) (#2482) (#2689) (#2655)
ADSOUT:    #515C (#2798) (#2816)
ADSOUTB:   #5188 (#2383) (#2417) (#2428) (#2433) (#2435) (#2446) (#2447) (#2465) (#2467) (#2479)
           (#2481) (#2496) (#2497) (#2525) (#2527) (#2541) (#2543) (#2556) (#2558) (#2566)
           (#2575) (#2577) (#2586) (#2592) (#2595) (#2685) (#2624) (#2625) (#2652)
ADSOUTT:   #5118 (#2385) (#2567) (#2587) (#2684) (#2656)
ADSQMAT:   #5186 (#2376) (#2441) (#2451) (#2486) (#2584) (#2511) (#2512) (#2534) (#2651)
ADAP:      ###7D (#1958) (#1968)
ADAP1:     ####8# (#132#) (#1347)
ADAP2:     ###93 (#1367) (#1374)
ADIN:      #514C (#2761) (#2772) (#2794)
ADOUT:     #514F (#2749) (#2797)
AFDT8:     ##8E8 (##44#) (##463) (#1#41) (#1735)
AWAIT:     ###13 (##529) (##531)
BASDECB:   #5482 (#3213) (#3317) (#3598) (#3831)
BASDECT:   #548# (#3228) (#364#) (#37#1) (#383#)
BASFCB:    #54B4 (#2925) (#3151) (#3169) (#32#7) (#3383) (#3832)
BASICB:    #54AC (#2989) (#3#47) (#3246) (#326#) (#3276) (#3288) (#33#8) (#3442) (#3458) (#3477)
           (#3495) (#3568) (#3589) (#3633) (#3694) (#3828)
BASOQB:    #54AE (#2963) (#2979) (#3#55) (#3#62) (#387#) (#3#77) (#3#99) (#318#) (#3191) (#3197)
           (#3399) (#3412) (#3545) (#3738) (#3753) (#3769) (#3782) (#3829)
BASES:     #58FF (#2376) (#2382) (#2391) (#2399) (#2641)
BCTSBA:    ##582 (###57) (#2642) (#2748) (#276#) (#2771) (#2961) (#2987) (#3177) (#3211) (#3226)
           (#3397) (#341#)
BUSIS:     ####1 (##437) (#3856)
BWAIT:     ###2A (#21#5) (#21#6)
CKEND:     ###85 (#1834) (#1974) (#2##8)
CKP:       ###2# (##812) (##83#)
CLOSE:     ###5A (##912) (##924)
CON1S:     #48A4 (##359) (##471)
CON2S:     #4AF8 (##971) (#1#49)
CON3S:     #4D84 (#1664) (#1743)
CONT1:     ###24 (##832) (##835)
CONT2:     ###4E (#1262) (#1264)
CONT2S:    #4858 (##217) (##221)
CONT3S:    #4827 (##145) (##149)
COUNT:     ###34 (#21#5) (#2123)
CSPUSNOS:  #21FC (##439) (##467) (#1#45) (#1739)
DASRC:     #4ED3 (##153) (##157) (##352) (#2#41)
DASTX:     #4A21 (##2#4) (##2#8) (##287) (##792)
DCS2#:     #533# (#3392) (#34#7)
```

```
  DCB21:     0533E (03404) (03421)
  DCB30:     05347 (03431) (03479)
  DCB31:     05353 (03440) (03448)
  DCB32:     05361 (03456) (03465)
  DCB33:     05377 (03470) (03479)
  DCB50:     053A2 (03503) (03532)
  DCB51:     053B8 (03550) (03554)
  DCB60:     053BF (03566) (03574)
  DCB61:     053D1 (03584) (03604)
  DCB62:     053D4 (03587) (03595)
  DCB63:     053E8 (03600) (03608)
  DCB70:     053FC (03629) (03680)
  DCB71:     053FE (03627) (03631) (03644) (03664) (03673)
  DCB72:     05410 (03641) (03650)
  DCB73:     05424 (03642) (03670)
  DCB74:     05426 (03673)
  DCB75:     05428 (03646) (03666) (03677)
  DCB76:     0542B (03680)
  DCB77:     05433 (03692) (03722)
  DCB78:     05449 (03716) (03717)
  DCB79:     0544E (03702) (03722)
  DCB80:     05452 (03703) (03727)
  DCB81:     05455 (03651) (03731)
  DCB82:     05456 (03736)
  DCB83:     05461 (03744) (03745)
  DCB84:     05464 (03680) (03708) (03720) (03723) (03729) (03739) (03750)
  DCB86:     05475 (03756) (03764)
  DCB87:     05478 (03769) (03771)
  DCB89:     05481 (03760) (03765) (03775)
  DCB90:     054A6 (03340) (03373) (03525) (03819)
  DCDRS:     05162 (00077) (02925)
  DECB10:    052E6 (03217) (03316)
  DECB20:    053DF (03219) (03597)
  DECDS:     05314 (02933) (03383)
  DECT20:    05407 (03232) (03639)
  DECT21:    0543C (03233) (03700)
  DMYS:      00794 (00436) (00910) (00911) (00913) (00914)
  ENS100:    050CD (02586)
  ENS101:    050D5 (02592) (02600)
  ENS102:    050E4 (02568) (02588) (02604)
  ENS111:    050E2 (02597) (02600)
  ENS30:     04FF6 (02412) (02416)
  ENS40:     05005 (02429) (02437)
  ENS50:     05010 (02446) (02448)
```

```
EN051:    0501F (02442) (02451)
EN060:    05028 (02461) (02469)
EN070:    0503D (02476) (02483)
EN080:    0504A (02475) (02486) (02505) (02535)
EN081:    05054 (02488) (02491) (02549)
EN082:    05056 (02492) (02502)
EN083:    0506B (02489) (02509)
EN084:    0506E (02511) (02516)
EN085:    0507D (02522) (02532)
EN086:    05092 (02515) (02539)
EN087:    05095 (02541) (02547)
EN088:    050A3 (02513) (02553)
EN089:    050A6 (02555) (02562)
EN090:    050B2 (02490) (02566)
EN091:    050BC (02572) (02582)
ENCD0:    04FC2 (00076) (02363)
ENDCK:    00070 (02126) (02220)
F10:      00001 (00063)
FCB:      0006E (00074) (00075)
FCB240:   000F0 (00462) (00463)
FCB242:   000F2 (01040) (01041)
FCB243:   000F3 (01734) (01735)
FDTSUFCB: 000C4 (00058) (00075)
FEND:     000A2 (01381) (01406) (01408)
FILTER:   0002A (00047)
FINISH:   0006F (00090) (00953)
GAP:      00020 (01031) (01033)
GAP1:     0000C (01110) (01112)
ICB10:    051DD (02993) (03046)
ICB11:    05297 (02994) (03245)
ICB12:    052A9 (02995) (03259)
ICB13:    052C8 (02996) (03287)
ICB14:    052DD (02997) (03307)
ICB20:    05355 (02999) (03441)
ICB21:    05363 (03000) (03457)
ICB22:    05376 (03001) (03476)
ICB23:    05383 (03002) (03494)
ICB24:    053C1 (03003) (03567)
ICB25:    053D6 (03004) (03588)
ICB26:    05400 (03005) (03632)
ICB27:    05435 (03006) (03693)
IIDSTB:   05112 (02364) (02424) (02455) (02657)
INS20:    05176 (02951) (02952)
INS40:    05193 (02979) (02980)
```

PAGE 100: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```
  INSCST:    04A8F (00277) (00878)
  INSCSZ:    04A9F (00279) (00895)
  INSFBS:    04A6B (00275) (00851)
  INSFTH:    04A57 (00273) (00834)
  INSGSZ:    04AF1 (00281) (00955)
  INSGTH:    04A4F (00271) (00829)
  INSRC:     04F1B (00284) (02107)
  INSTX:     04CB1 (00282) (01500)
  IN1:       0004A (00905) (00908)
  IN2:       00047 (00894) (00954) (00958)
  IN4:       00043 (00890) (00892) (00916) (00945)
  INDSIN:    0515E (02017)
  INDSOUT:   0515F (02751) (02789) (02807) (02818)
  INIS:      0485C (00081) (00269)
  INITS:     05170 (02944)
  ISVTS:     00502 (00059) (00064) (00065) (00066) (00067) (00068) (00130) (00203) (00270) (00347)
             (02738) (02756) (02767) (02784) (02932) (02945) (03141) (03143) (03145) (03347)
             (03367) (03369) (03386) (03517) (03519) (03521) (03810) (03812) (03822)
  ISVT1S:    00503 (00064) (00136) (00206) (00272)
  ISVT2S:    00504 (00065) (00138) (00219) (00274) (00350)
  ISVT3S:    00505 (00066) (00147) (00276)
  ISVT4S:    00506 (00067) (00152) (00278)
  ISVT5S:    00507 (00068) (00280)
  LSRUN:     05115 (00295) (02509) (02539) (02553) (02660) (03653) (03661) (03713)
  LOOP0:     00030 (00586) (00619) (00634)
  LOOP1:     0001F (00544) (00553)
  LOOP2:     00005 (00498) (00511)
  LOOP4:     000AC (00728) (00754)
  LOOP5:     00098 (00730) (00751) (00756)
  LOOP6:     00034 (00591) (00608)
  LOOP7:     0004E (00612) (00621)
  LOOP8:     00058 (00625) (00633)
  LOOP9:     00068 (00655) (00678)
  MBS:       00000 (00435) (00787) (00788) (00930) (01487) (01488) (01557) (02044) (02045) (02059)
             (02089) (02090)
  MSKSRBVT:  000FF (00062) (00201) (00269) (00346)
  NSRC:      00466 (00134) (00355) (00456) (00457)
  NSTX:      00430 (00202) (00286) (00455)
  NEGA:      00087 (01286) (01433)
  NEXT1:     00027 (00836) (00838)
  NOSMAX:    00009 (02897) (02950) (03023) (03025) (03108)
  NOFTR:     0003D (00839) (00867) (00875)
  NULL1:     0009E (01404)
  OQB01:     05262 (03184) (03190)
```

```
  OOB02:    0526A (03185) (03196)
  OOB10:    05194 (02967) (02978)
  OOB11:    051E2 (02968) (03054)
  OOB12:    051EA (02969) (03061)
  OOB13:    051F0 (02970) (03069)
  OOB14:    051F8 (02971) (03076)
  OOB15:    05206 (02972) (03098)
  OOB26:    0547C (03422) (03760)
  OOB27:    05487 (03423) (03781)
  PSBASE:   05488 (00299) (03036) (03125) (03128) (03358) (03361) (03507) (03510) (03798) (03801)
                  (03836)
  PSBIT1:   05489 (03135) (03238) (03255) (03272) (03280) (03429) (03618) (03792) (03795) (03837)
  PSINFO:   054BA (03331) (03617) (03624) (03788) (03838)
  PSMAX:    054BB (02951) (03026) (03027) (03094) (03103) (03109) (03839)
  PSSHAT:   05487 (03203) (03552) (03751) (03775) (03835)
  PAR81:    050F1 (02622) (02627)
  PAROUT8:  050EF (02508) (02530) (02546) (02561) (02580) (02598) (02621)
  PCRC8:    04D86 (01737) (01751)
  PCRC8SA:  00000 (01748) (01764) (02010)
  PCRC8SZ:  0009A (01749) (02018)
  PCTX8:    04AFA (01043) (01057)
  PCTX8SA:  00000 (01054) (01079) (01444)
  PCTX8SZ:  000C1 (01055) (01444)
  PERIOD:   00029 (00530) (00566)
  PICH38:   04BA6 (00465) (00479)
  PICH38SA: 00000 (00476) (00493) (00760)
  PICH38SZ: 0000A (00477) (00760)
  PRE:      0002C (01049) (01981)
  PRE1:     0003F (01071) (01077)
  PRE2:     0002A (01165) (01171)
  PRE3:     00011 (01105) (01113) (01136) (01382)
  QCKP:     0001D (01147) (01149)
  QUAN:     00046 (01092) (01093) (01097)
  QUAN1:    00052 (01272) (01273) (01280)
  QUAN2:    0005D (01283) (01442)
  QUAN3:    00065 (01294) (01439)
  QUAN4:    00075 (01301) (01311)
  QUAN5:    0007B (01310) (01314) (01317)
  QUAN6:    0007F (01278) (01441)
  RC8:      04000 (00079) (00126)
  RCINI8:   04090 (00002) (00346)
  RCP1:     000AF (01197) (01198) (01422)
  RECORD:   00068 (00891) (00942)
  RESET:    000A9 (00702) (00750)
```

PAGE 102: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1980

```
  $STX:      04A1D (00213) (00222) (00289) (00789)
  SC810:     051C5 (03026) (03029)
  SC820:     051DA (03045) (03063)
  SC821:     051E9 (03058) (03062)
  SC822:     051EE (03050) (03068)
  SC823:     051F7 (03073) (03077)
  SC824:     051F9 (03064) (03001)
  SC831:     052D8 (03092) (03096) (03101) (03103)
  SC841:     052DF (03109) (03111)
  SC851:     0521C (03110) (03121)
  SC852:     05261 (03191) (03192)
  SC853:     05295 (03244) (03250)
  SC854:     052A5 (03257) (03266)
  SC8541:    052BE (03270) (03280)
  SC855:     052C6 (03286) (03293)
  SC856:     052D8 (03303) (03321)
  SC857:     052D8 (03306) (03314)
  SC858:     052EF (03310) (03327)
  SC860:     052F8 (03345)
  SC870:     052FD (03110) (03353)
  SHSRC:     04F0F (00141) (00151) (00349) (02092)
  SHIFT:     00051 (01507) (01590)
  SIDSISL:   05486 (02931) (03139) (03345) (03365) (03384) (03515) (03808) (03834)
  SIDSTB:    05485 (03403) (03417) (03532) (03833)
  SIGNAL:    00098 (01029) (02007)
  SIZES:     0516D (02775) (02819)
  START:     00008 (01497)
  SVTS:      00302 (00060) (00128) (00442) (02426) (02458) (03161) (03165) (03535) (03541)
  SVT111S:   00464 (00454) (00456) (02231)
  SVT115S:   00468 (00457) (00458) (02038)
  SVT117S:   0046C (00460) (00459) (00784)
  SVT118S:   0046E (00459) (01485)
  SVT50S:    003E6 (00442) (00443) (00783)
  SVT51S:    003E8 (00443) (00444) (00847)
  SVT54S:    003EE (00444) (00445) (00876)
  SVT56S:    003F2 (00445) (00446) (00924)
  SVT63S:    00404 (00446) (00447) (01480) (01643)
  SVT68S:    0049E (00447) (00448) (01506)
  SVT70S:    00412 (00448) (00449) (01529)
  SVT74S:    0041A (00449) (00450) (01550)
  SVT79S:    00426 (00450) (00451) (01603)
  SVT83S:    0042E (00451) (00452) (00455) (01642)
  SVT98S:    00446 (00452) (00453) (02040)
  SVT99S:    0044A (00453) (00454) (02233)
```

```
SYNCS:    0510C (02934) (03023)
SYSFLGS:  1FFCE (00061) (00143) (00146) (00215) (00218) (00353)
SZSHAT:   0054D6 (00201) (00375) (03847)
TSLIM:    0054D7 (03848)
TSRC:     004F13 (00132) (02095)
TOESNEW:  0054D0 (03854) (03856)
TOESPRT:  00208 (00438) (03855)
TRS10:    0512A (02740) (02755)
TRS20:    05135 (02739) (02766)
TRS25:    0513F (02762) (02775)
TRS30:    05148 (02795) (02004)
TRSDC:    05116 (00078) (02735)
TRAN:     00000 (01772) (01783)
TXS:      04836 (00080) (00201) (03823)
VSBETA:   0054D1 (03327) (03540) (03608) (03842)
VSDEL:    05114 (00297) (02474) (02608) (02659)
VSGAP:    0054D5 (03012) (03551) (03846)
VSPAR:    0054D3 (03336) (03489) (03501) (03844)
VSPHS:    0054D2 (03300) (03319) (03546) (03581) (03602) (03843)
VSSYN:    05113 (02411) (02418) (02668)
VSSYNC:   0054CF (03042) (03120) (03354) (03492) (03805) (03840)
VST:      0054D0 (03295) (03534) (03576) (03841)
VSTHR:    0054D4 (03650) (03707) (03845)
VADAP:    00058 (02192)
VCHECK:   00072 (01505) (01521) (01633) (01635)
VCONT1:   00045 (01567) (01568) (01578)
VEND:     00075 (01634) (01641)
VGAP:     0001A (01513) (01520)
VHSRC:    04F2D (00133) (00140) (00348) (02124)
VHSTX:    04A25 (00209) (00212) (00288) (00795)
VLOOP:    00023 (01529) (01633)
VLOOP1:   00035 (00086) (00866)
VLOOP2:   0000E (00799) (00811)
VLOOP3:   0001A (00798) (00811)
VLOOP4:   0001C (00813) (00816)
VLOP1:    0002D (02109) (02115)
VLOP2:    00071 (02221) (02223)
VNEGA:    00058 (01591) (01593) (01594)
VPCRCS:   04EC2 (01730) (02019) (02026) (02252)
VPCRCSA:  04F06 (02022) (02242)
VPCRCSI:  04FBE (02018) (02250)
VPCRCSS:  00013 (02519) (02059)
VPCRCSZ:  00100 (02021) (02252)
VPCTXS:   04C04 (01044) (01453) (01459) (01662)
```

PAGE 104: MAP MODULES FOR THE P A R C ALGORITHM --- JAN. 30, 1988

```
  VPCTX0A:   04C02 (01456) (01662)
  VPCTX0I:   04D02 (01452) (0166D)
  VPCTX0S:   00030 (01453) (01557)
  VPCTX0Z:   00100 (01455) (01662)
  VPICH30:   04A0E (00466) (00767) (00773) (00969)
  VPICH30A:  04A16 (00770) (00960)
  VPICH30I:  04AF6 (00766) (00967)
  VPICH30S:  00060 (00767) (00930)
  VPICH30Z:  000EA (00769) (00969)
  VPRE1:     00037 (02134) (02222)
  VQ1:       0004C (02166) (02166) (02169)
  VQ3:       00061 (02165) (02170)
  VQUAN1:    00046 (01579) (01581) (01584)
  VQUAN2:    0004C (0158D) (01585)
  VQUAN3:    0004F (01579) (01588)
  VTRAN:     0000D (02047) (02052)
  WAIT:      00056 (00630) (00631)
  WRD4:      00003 (00069) (00126)
  WRD5:      00004 (00070) (00129)

LINES WITH ERRORS:    0  (MAP VERSION 00101.10)  E-    0
```

```
                (00001) ;
                (00002) ;                        ***   IOSDC   ***
                (00003) ;
                (00004) ;           ARVIND S. ARORA              FEB 11, 1980
                (00005) ;
                (00006) ;      IOS-2 PROGRAM TO INTERFACE THE BITSTREAM FROM THE PARC ALGORITHM
                (00007) ; TO THE MODEM. THE SOURCE CODER AT THE TRANSMITTER STORES THE OUTPUT
                (00008) ; IN A DOUBLE BUFFER, AND THE DECODER REQUIRES ITS INPUT TO BE IN A
                (00009) ; CIRCULAR BUFFER. THIS PROGRAM OUTPUTS THE CONTENTS OF THE DOUBLE
                (00010) ; BUFFERS THROUGH THE IOS2 TO THE DIGITAL INTERFACE, AND INPUTS THE
                (00011) ; RECEIVED BITSTREAM OF THE DIGITAL INTERFACE TO THE CIRCULAR BUFFER.
                (00012) ; THE PROGRAM OPERATES IN FULL DUPLEX FOR COMPATIBILITY WITH THE
                (00013) ; RS-423 MODEM USED IN THE PRESENT SYSTEM.
                (00014) ;
                (00015) ;
                (00016) ;     REGISTER & FLAG USAGE:
                (00017) ;
                (00018) ;  R0: XMTR BUFFER SWITCH, R0=ADSTXA  TO INDICATE BUFFER 1
                (00019) ;                         R0=ADSTXB  TO INDICATE BUFFER 2
                (00020) ;  R1: XMTR BUFFER ADDRESS POINTER
                (00021) ;  R2: RCVR BUFFER ADDRESS POINTER
                (00022) ;
                (00023) ;  F1: FLAG TO INDICATE MODE OF OPERATION,
                (00024) ;                         F1=CLEAR  TO INDICATE INITIALIZE
                (00025) ;                         F1=SET    TO INDICATE NORMAL OPERATION
                (00026) ;  P1: FLAG SET INDICATES RECEIVER DATUM AVAILABLE
                (00027) ;  P2: FLAG SET INDICATES XMTR READY FOR NEXT DATUM
                (00028) ;
                (00029) ;
                (00030) ;------ IOS-2 MODULE
                (00031) ;
                (00032) ; ---     SYMBOL DEFINITIONS
                (00033) ;
00006464        (00034)           ADSTXA=D'25700'
00006C34        (00035)           ADSTXB=D'27700'
000000BD        (00036)           SZSTX =D'189'
000065B2        (00037)           ENDSTXA=ADSTXA+SZSTX-1
00006CF8        (00038)           ENDSTXB=ADSTXB+SZSTX-1
                (00039) ;
00007148        (00040)           ADSRCC=D'29000'
00000400        (00041)           SZSRC=D'1024'
00007547        (00042)           ENDSRCC=ADSRCC+SZSRC-1
                (00043) ;
0000ECC4        (00044)           CLCRATE=$ECC4
```

PAGE 2:

```
                    (00045) ;
                    (00046) ;
          00000000  (00047) #L=0
                    (00048) ;
                    (00049) ; ---   1) LOAD CLOCK RATE
                    (00050) ;
                    (00051) ;
                    (00052)         BEGIN  IOS2(IOSDC)
A00 00000 01000CC4  (00053) IOSDC0  LOAD(R0,CLCRATE)                 CLOCK RATE
A01 00002 02300200  (00054)         MR
A02 00004 03390400  (00055)         ADDL(R0,0,TP)                    LOAD CLCK. RATE
A03 00006 04300000  (00056)         STOP
                    (00057) ;
                    (00058) ; ---   2) INITIALIZE BASE ADDRESSES
                    (00059) ;
A04 00008 05420463  (00060) IOSSINIT LOAD(R1,ADSTXA-1(1),L)          INIT. XMTR ADDR
A05 0000A 06027147  (00061)         LOAD(R2,ADSRCC-1(1),L)           INIT. RCVR ADDR
A06 0000C 07302400  (00062)         INT1
                    (00063) ;
                    (00064) ; ---   3) LOOP TO CHECK DATA AVALAIBILTY
                    (00065) ;
A07 0000E 08341C00  (00066) IOSSLOOP JIFS(TRX0,P2)                   GO TO XMTR LOOP IF P2
A08 00010 0C100000
A09 00012 0A340C00  (00067)         JIFS(RCC0,P1)                    GO TO RCVR LOOP IF P1
A0A 00014 1F100000
A0B 00016 07300400  (00068)         JUMP(IOSSLOOP)
                    (00069) ;
                    (00070) ; ---   4) TRANSMITTER LOOP
                    (00071) ;
A0C 00018 0D300600  (00072) TRX0    MR                               TRANFER MAP --> MODEM
A0D 0001A 0E2C6C34  (00073)         JEQ(TXB0,R0,ADSTXB)              CHECK BUFF 1 OR 2
A0E 0001C 17100000
                    (00074) ;
                    (00075) ; ---   BUFFER 1
                    (00076) ;
A0F 0001E 1F5A0001  (00077) TXA0    ADD(R1,1,TM)                     INCR ADDR & INIT TRANF.
A10 00020 12606520  (00078)         JNE(IOSSLOOP,R1,ENDSTXA)         CHECK BUFFER FILL STAT.
A11 00022 00000000
A12 00024 07100000
                    (00079) ;
A13 00026 14304000  (00080)         INT2
A14 00028 15006C34  (00081)         LOAD(R0,ADSTXB)                  FULL, MOVE TO BUFF 2
A15 0002A 16426C33  (00082)         LOAD(R1,ADSTXB-1(1),L)
A16 0002C 07300400  (00083)         JUMP(IOSSLOOP)
```

```
                        (####4) ;
                        (####5) ; ---   BUFFER 2
                        (####6) ;
  A17 ###2E 185A###1 (####7) TXBS        ADD(R1,1,TM)                    SAME AS BUFFER 1
  A18 ###3# 1A686CF# (####8)             JNE(IOS$LOOP,R1,END$TXB)
  A19 ###32 ########
  A1A ###34 #71#####
                        (####9) ;
  A1B ###36 1C3#2### (###9#)             INT1
  A1C ###38 1D##6464 (###91)             LOAD(R#,ADSTXA)
  A1D ###3A 1E426463 (###92)             LOAD(R1,ADSTXA-1(1),L)
  A1E ###3C #73##4## (###93)             JUMP(IOS$LOOP)
                        (###94) ;
                        (###95) ; ---   5) RECEIVER LOOP
                        (###96) ;
  A1F ###3E 2#3##2## (###97) RCC$        MW                              TRANSFER MODEM --> MAP
  A2# ###4# 219A###1 (###98)             ADD(R2,1,TM)                    INCR ADDR & INIT TRANF.
  A21 ###42 22A87547 (###99)             JNE(IOS$LOOP,R2,END$RCC)        CHECK BUFFER FULL STATUS
  A22 ###44 #71#####
  A23 ###46 24827147 (##1##)             LOAD(R2,ADSRCC-1(1),L)          FULL, RE-INIT. BASE ADDR
  A24 ###48 #73##4## (##1#1)             JUMP(IOS$LOOP)
                        (##1#2) ;
                        (##1#3) ; ---   STOP IF FLAG F2 SET
                        (##1#4) ;
  A25 ###4A 263#6### (##1#5) IOS$STOP    STOP
      ###4C            (##1#6)           END
```

PAGE 4:

```
  ADSRCC:    #7148 (####4#) (###42) (###61) (#1##)
  ADSTXA:    #8464 (###34) (###37) (###6#) (###91) (###92)
  ADSTXB:    #8C34 (###35) (###38) (###73) (###81) (###82)
  CLCRATE:   #ECC4 (###44) (###63)
  ENDSRCC:   #7647 (###42) (###99)
  ENDSTXA:   #852# (###37) (###78)
  ENDSTXB:   #8CF# (###38) (###88)
  IOSSINIT:  ####4 (###6#)
  IOSSLOOP:  ####7 (###66) (###68) (###78) (###83) (###86) (###93) (###99) (##1#1)
  IOSSSTOP:  ###25 (#1#5)
  IOSDCS:    ##### (###53)
  RCCS:      ###1F (###67) (###97)
  SZSRC:     ##4## (###41) (###42)
  SZSTX:     ###0D (###36) (###37) (###38)
  TRXS:      ####C (###66) (###72)
  TXAS:      ####F (###77)
  TXBS:      ###17 (###73) (###87)

LINES WITH ERRORS:    #  (MAP VERSION 8##1#1.1#)   E-    #
```